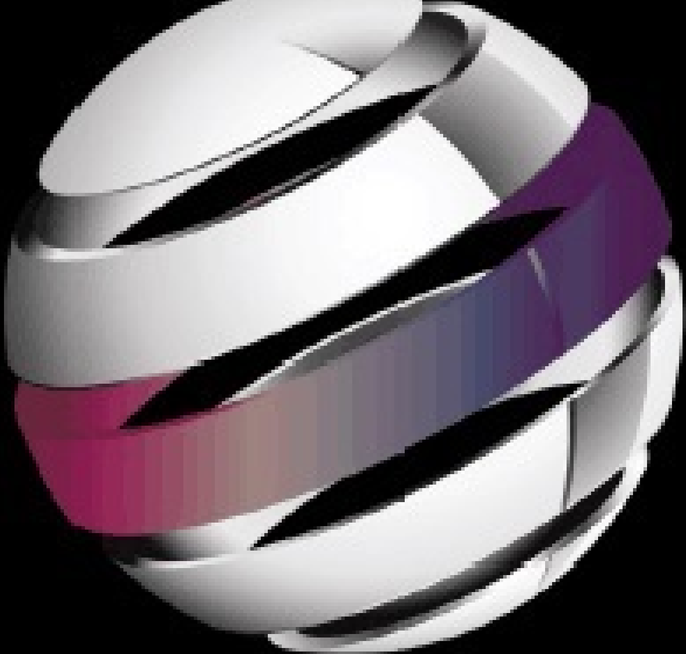


Learn Android Studio

Build Android Apps Quickly and Effectively

Adam Gerber | Clifton Craig

Studio

Apress®

Table of Contents

Android Studio 中文教程（前六章）

> 译者：Android Studio 中文社区
>
> 整理：飞龙

第一章：Android Studio 介绍

> 译者：EvilSong
>
> 来源：Learn Android Studio 汉化教程 第一章 : Android Studio 介绍

本章将引导您完成安装和设置开发环境,然后你就可以跟随本书的例子和课程学习。首先,您将安装被称为Java开发工具包(JDK)的必要组件。然后你要下载和安装Android Studio以及Android软件开发工具包(SDK),这些都是开发Android应用程序所必需的工具。接着，我们将向你展示如何使用新建项目向导来创建一个简单的项目HelloWorld。最后，我们将向你展示怎样连接Android虚拟设备（AVD）和Android物理设备。在本章结束的时候，你将在Android Studio中拥有开发Android应用程序的一切东西。

Windows下安装JDK

本节适用于Windows用户。如果你是Mac用户，请跳转到下一节“在Mac上安装JDK”。Android Studio 使用java编译环境构建，因此在开始使用Android Studio之前你需要确保你已经安装Java开发工具包（JDK）。如果你是一个经验丰富Android或Java开发人员的话，你很有可能已经在你的电脑上安装了JDK。如果你的电脑上的JDK运行版本是1.6或更高的话，你可以跳过这一节。此外，你将下载，安装和配置最新版本的JDK。你可以从下面的Oracle网站上下载JDK：

```
http://www.oracle.com/technetwork/java/javase/downloads/index.html
```

打开网站之后，点击java Download 按钮，如图1-1所示。

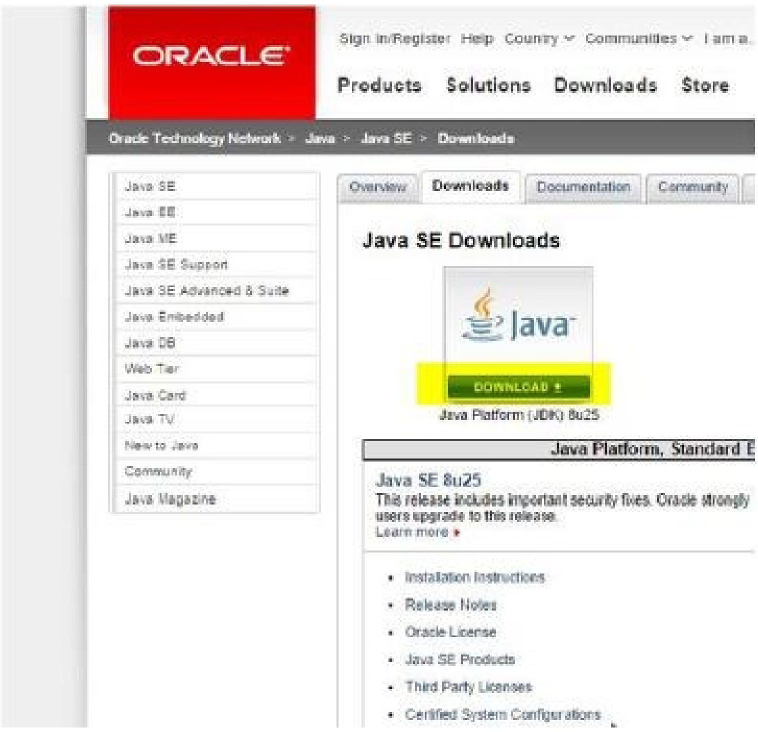


图1-1 Java下载界面上的Java Download 按钮

下载Windows上使用的JDK

下一步如图1-2所示，需要单击单选按钮来接受许可证协议。然后选择适合你操作系统的JDK。如果你是Windows 7或者Windows 8系统，你应该点击文件链接右边窗口的x64版本，如图1-2所示。Oracle会频繁的发布JDK更新版本。在本书印刷之际，新版本的JDK基本能够使用了，所以请下载最新版本的JDK。然后等待安装文件下载。文件大小通常约125MB，所以下载应该花不了多长时间。

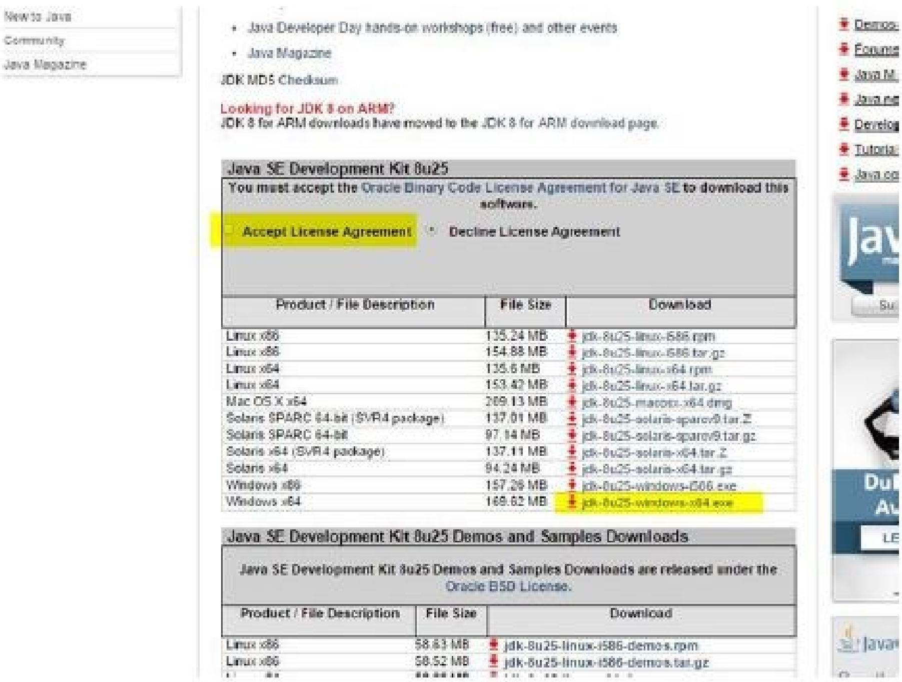


图1-2 接受许可协议，点击适当的链接

在Windows上执行JDK向导

安装JDK之前，在你C盘的根目录下新建一个名为Java的文件夹。文件夹的名字可以是任意的，之所以称为Java是因为有许多跟Java相关的工具要安装在这里，包括JDK，Android Studio和Android SDK。将Android Studio相关的工具安装在C:\Java目录下能够保证你的开发环境更有条理。 找到你浏览器下载安装文件的位置然后双击执行。当安装开始，你将看到安装向导，如图1-3所示。在Windows系统下，JDK默认安装在 `C:\Program Files\Java\` 目录下。要更改安装目录的位置，请单击Change按钮。我们建议安装JDK字 `C:\Java` 目录下是因为它的路径名中不包含空格，很容易记住。如图1-4所示。



图1-3 JDK安装向导窗口

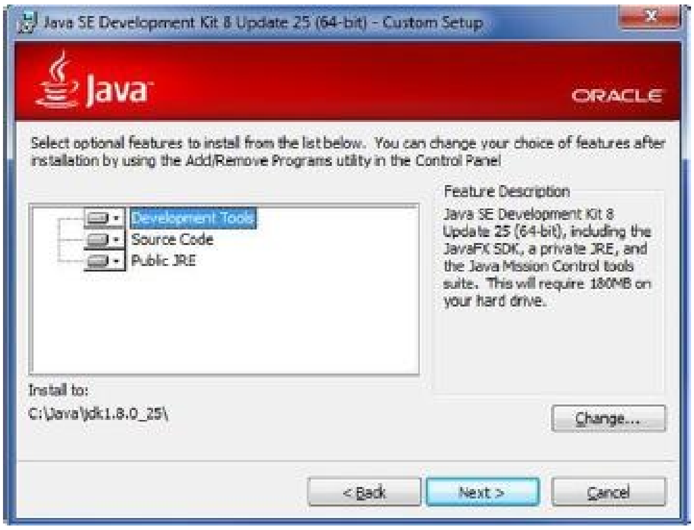


图1-4 选择JDK安装目录

记住JDK的安装位置。按照提示，直到安装完成。如果提示安装Java运行环境（JRE），选择跟JDK安装位置相同的目录。

Windows上配置环境变量

本节将向你展示如何配置Windows以便Android Studio找到JDK。在电脑上运行Windows，按住Windows键和Pause键打开系统窗口。单击高级系统设置选项，如图1-5所示。

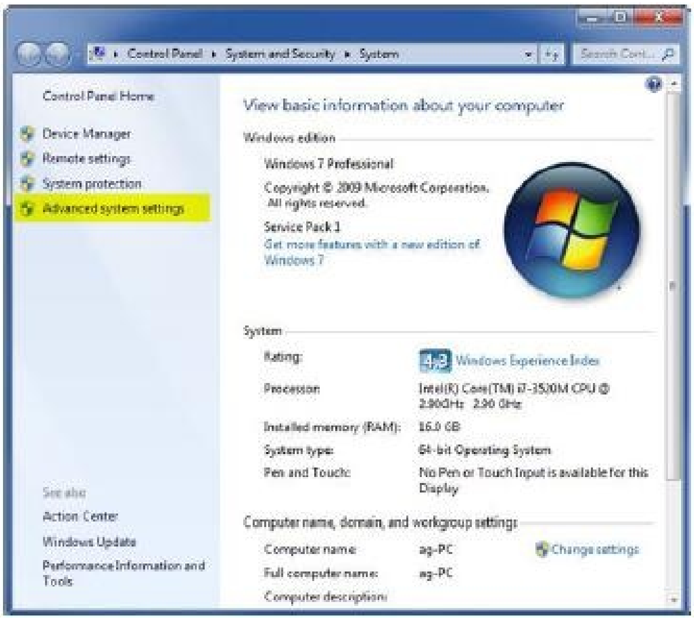


图1-5 Windows系统窗口

点击“环境变量”按钮，如图1-6所示。如图1-7所示，在下面的系统变量列表中找到 `JAVA_HOME` 项。如果 `JAVA_HOME` 不存在，点击新建来创建，否则，点击编辑。

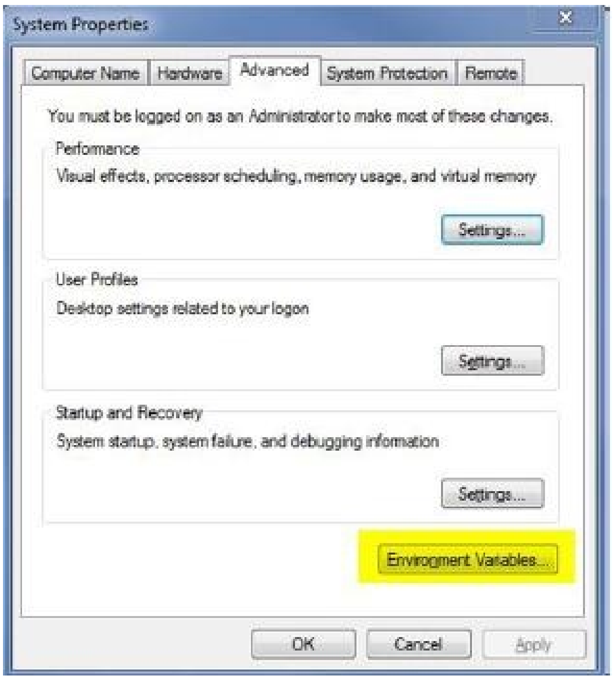


图1-6 系统属性

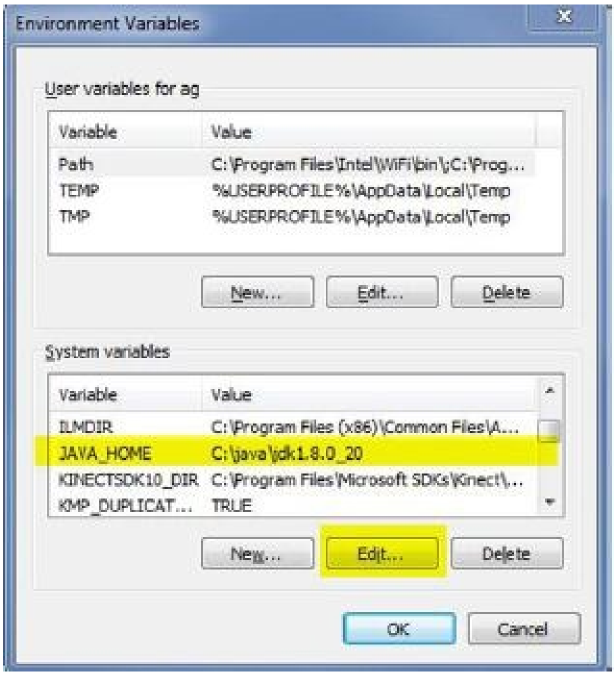


图1-7 环境变量

点击新增或者编辑显示一个对话框如图1-8所示。务必在变量名中输入 `JAVA_HOME` 。然后在变量值中输入刚才JDK的安装位置（去掉后面斜杠），如图1-4所示。然后点击确定。



图1-8 编辑 `JAVA_HOME` 环境变量

同样，就像编辑JAVA_HOME环境变量一样，你需要编辑 `PATH` 环境变量。如图1-9所示。将你的光标移动到变量值的最后面然后输入如下路径：

```
;%JAVA_HOME%\bin
```



图1-9 编辑PATH环境变量

然后点击确定，确定，确定，保存这些更改并退出系统属性。

为了测试新的JDK安装成功，单击开始按钮，输入 `cmd` ，然后回车，打开命令行窗口。在命令行窗口中输入以下命令并按回车：

```
java -version
```

如果你有如图1-10所示的信息，恭喜你，你已经成功安装了JDK。

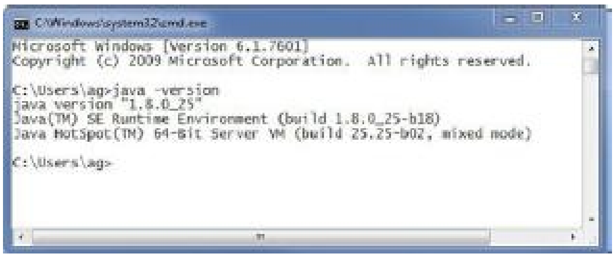


图1-10 确认JDK安装成功

在Mac上安装Java开发工具包

Mac和Windows安装JDK的前两个步骤是相同的。用浏览器打开如下网站：

```
http://www.oracle.com/technetwork/java/javase/downloads/index.html
```

在界面上点击Java Download按钮，如图1-11所示。

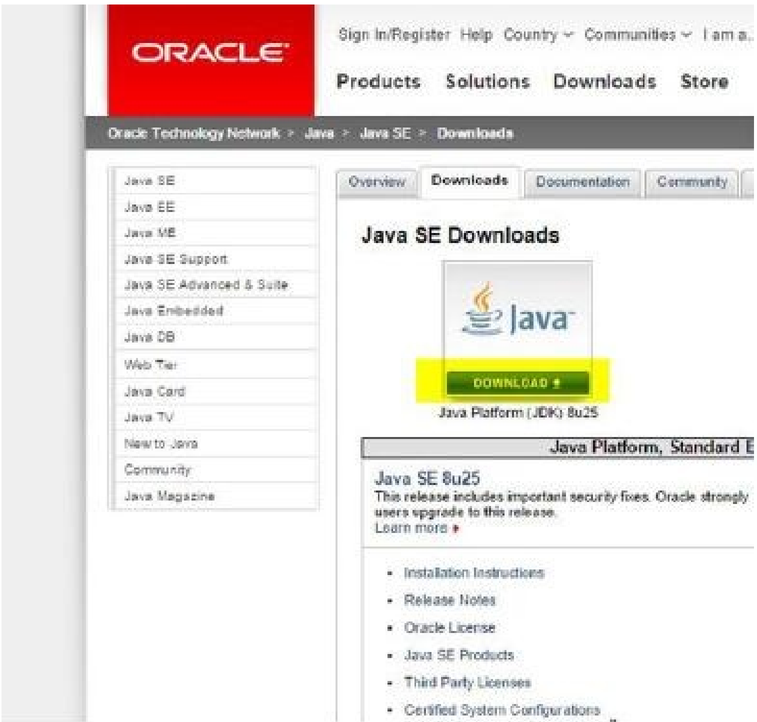


图1-11 Java Download页面上的Java Download按钮

下载Mac上使用的JDK

如图1-12所示，点击接受许可协议单选按钮接受许可协议。然后你必须选择适合你操作系统的JDK版本。如果你电脑是64位的OS X，你应该点击右侧的Mac OS X64标签，如图1-12所示。Oracle会频繁的发布JDK更新版本。在本书印刷之际，新版本的JDK基本能够使用了，所

以请下载最新版本的JDK。等待安装文件下载。

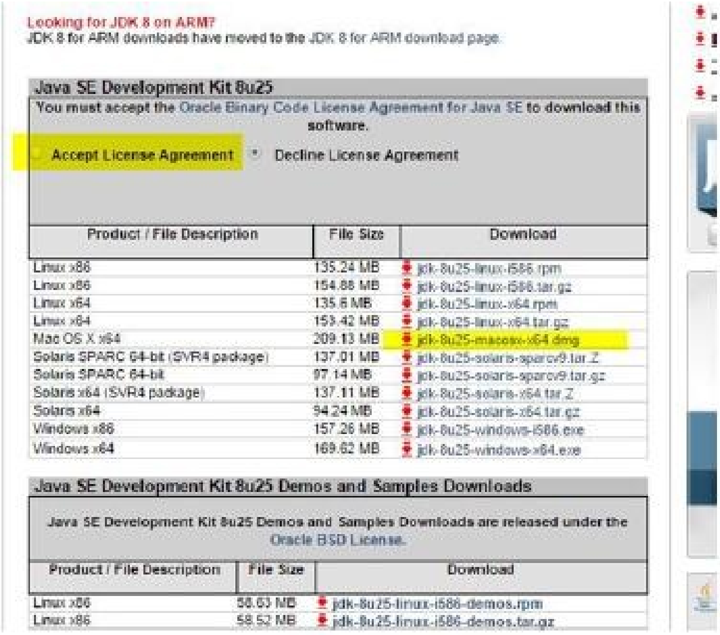


图1-12 接受许可协议，点击适当的链接

Mac上执行JDK向导

双击 `.dmg` 文件执行。然后点击 `.pkg` 文件打开向导然后根据需要继续。如图1-13到1-15。

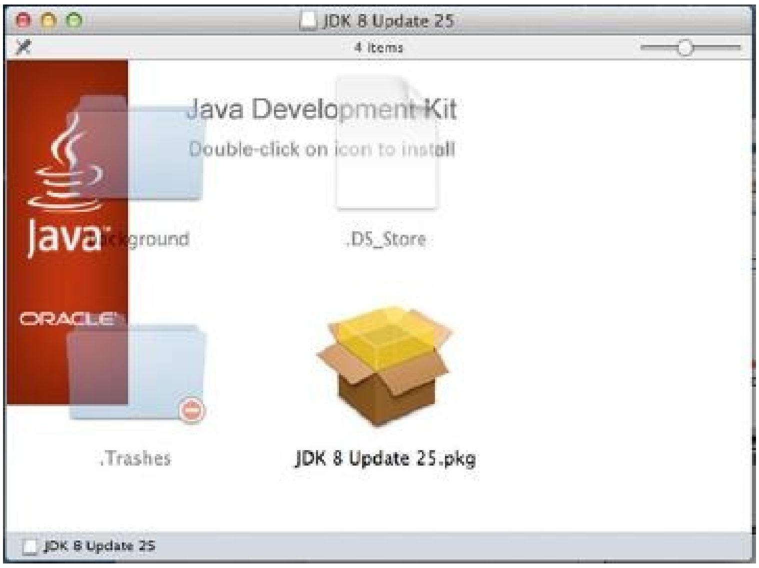


图1-13 `JDK 8 Update 25.pkg`

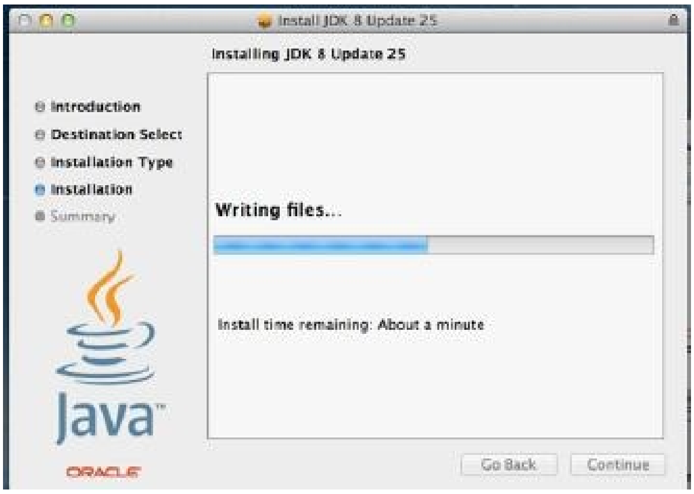


图1-14 安装向导



图1-15 安装成功

Mac上配置JDK版本

配置你的Mac以便Android Studio能够找到合适的JDK。打开Finder窗口然后选择应用➤实用工具。从那里打开Java偏好设置，按照指示将最新版本放置到列表的最上面作为首选版本。

安装Android Studio

在下载Android Studio之前，为本书中的课程创建一个根目录。我们使用 `C:\androidBook\` 作为本书的所有课程的根目录，但是你也可以选择或者创建任何你认为合适的目录。出于这个原因，我们就简单的成为课程根目录。

下载Android Studio 很简单，用浏览器打开如下网站：

```
http://ask.android-studio.org/?/developer.android.com/sdk/installing/studio.html
```

点击 `Download Android studio for you OS` 绿色大按钮，如图1-16所示。接着，勾选我已阅读并同意上述条款条件复选框。再次点击 `Download Android studio` 按钮，安装文件将开始下载。下载完成之后执行下载文件。



图1-16 下载Android Studio

安装向导开始之后，点击Next按钮一路向下直到选择组件窗口。勾选全部组件复选框如图1-17所示。然后点击Next。再次同意条款和条件。当你到达 `Configuration Settings:Install Locations` 窗口时如图1-18所示，选择Android Studio 和Android SDK的安装位置。为了保持一致，我们将Android Studio安装在 `C:\ Java \ astudio \` 下，将Android SDK安装在 `C:\ Java \ asdk \` 下。



图1-17 选择组件



图1-18 选择Android Studio和SDK安装位置

一路点击Next按钮直到Android Studio和Android SDK都安装完成。你最终显示的窗口应该如图1-19所示。勾选Start Android Studio复选框之后点击Finish按钮启动Android Studio。请确保选中复选框，然后点击Finish按钮，Android Studio才会启动。请注意从现在起，你需要从桌面图标或开始菜单中启动Android Studio。



图1-19 完成Android Studio设置

Android Studio第一次打开的时候，安装向导将查找你系统上的JDK（如前面安装的一个）和Android SDK的位置。安装向导应该为你的Android Studio 下载一切开始开发应用程序的东西。点击Finish按钮关闭安装向导。

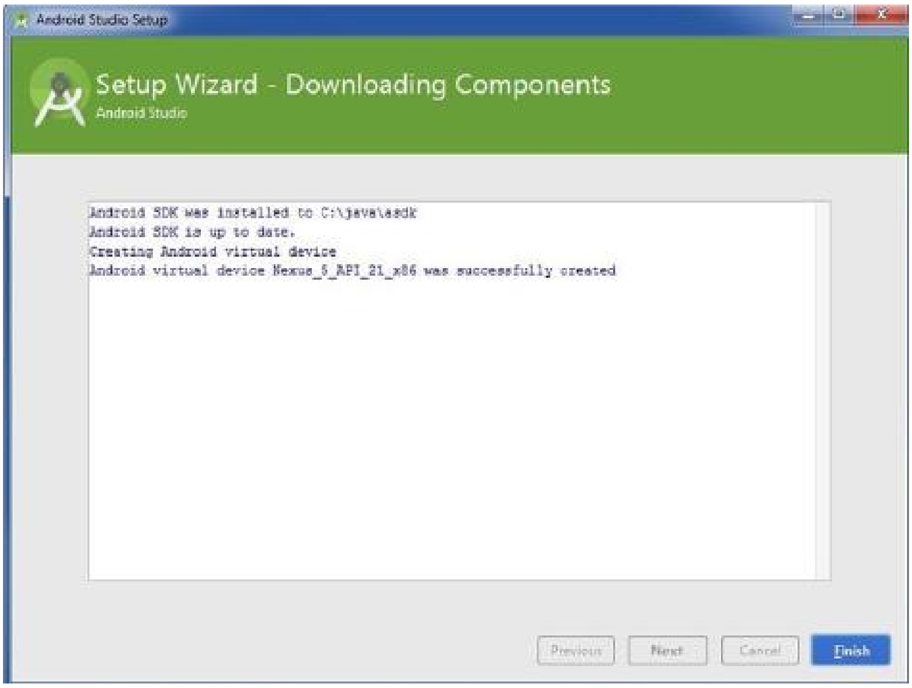


图1-20 安装向导-下载组件

新建你的第一个项目：HelloWorld

一旦安装向导完成，Android Studio的欢迎对话框将会出现，如图1-21所示。点击Start a new Android Studio project选项。



图1-21 欢迎来到Android Studio

在新建项目向导（如图1-22），在Application Name输入 `HelloWorld` ，在Company Domain中输入 `gerber.apress.com` 。请注意，Package name是反转的Company Domain加上Application Name。创建 `HelloWorld` 项目在课程的根目录文件夹下。正如前面提到的，如果是Windows系统，我们使用 `C:\androidBook\` 。如果是Mac或者Linux系统，你的根目录名称不是以字母开头，而是一个斜杠。



图1-22 配置你的新项目

Android操作系统可以在多种平台上运行，包括游戏机，电视机，手表，眼镜，智能手机和平板电脑。默认情况下，手机和平板电脑复选框会被选中，API-8被选为最低的SDK。同意这些设置然后点击Next，如图1-23所示。



图1-23 选择APP运行的形式

后续新项目向导将提示你选择一种布局。选择 `Blank Activity` 然后点击Next按钮。同意默认名字，如图1-24所示，他们应该如下：

```
Activity Name: MainActivity
Layout Name: activity_main
Title: MainActivity
Menu Resource Name: menu_main
```



图1-24 为新文件选择设置

使用Android虚拟设备管理器

Android虚拟设备管理器允许你创建Android虚拟设备（AVDs），然后你可以在你的电脑上运行模拟器。模拟和仿真有一个很重要但是微妙的区别。模拟意味着虚拟设备只有一个外形，模拟实际的物理设备如何运作，但是不针对特定的操作系统。IOS开发环境使用模拟器，对于有限数量的设备的平台的IOS来说可能是一个不错的选择。

然而对于仿真器而言，你的电脑留出一块内存去复制基于仿真器正在仿真设备上的环境。Android Studio使用仿真器，这意味着Android虚拟设备管理器启动一个 Linux内核的大沙箱和整个Android栈为了仿真基于Android物理设备的环境。尽管仿真器提供了一个比模拟器更可靠的环境来测试你的应用程序，但是启动一个AVD需要数分钟，这取决于你电脑的速度。好消息是你的仿真器仍然活跃在内存中，它仍然是有响应的。然而，如果你有Android手机或者平板电脑，我们建议使用物理设备来测试你的应用程序，而不是使用AVD。也就是说，我们首先使用Android虚拟设备管理器创建一个AVD，在后来的章节我们将想你展示如何连接你的物理设备，如果你有的话。

点击Android虚拟设备管理器图标如图1-25所示。在Android虚拟设备管理器向导的第一个界面中点击创建虚拟设备按钮。在下一个界面选择Galaxy Nexus如图1-26，然后点击Next。下一个界面允许你选择一个系统镜像如图1-27。选择第一个选项Lollopop（或最新的API）和x86_64的ABI。点击Next。在下一个界面上，点击Finish按钮来验证你的AVD设置。恭喜你，你已经创建了一个新的AVD。

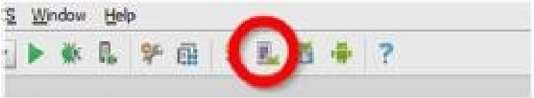


图1-25 AVD图标

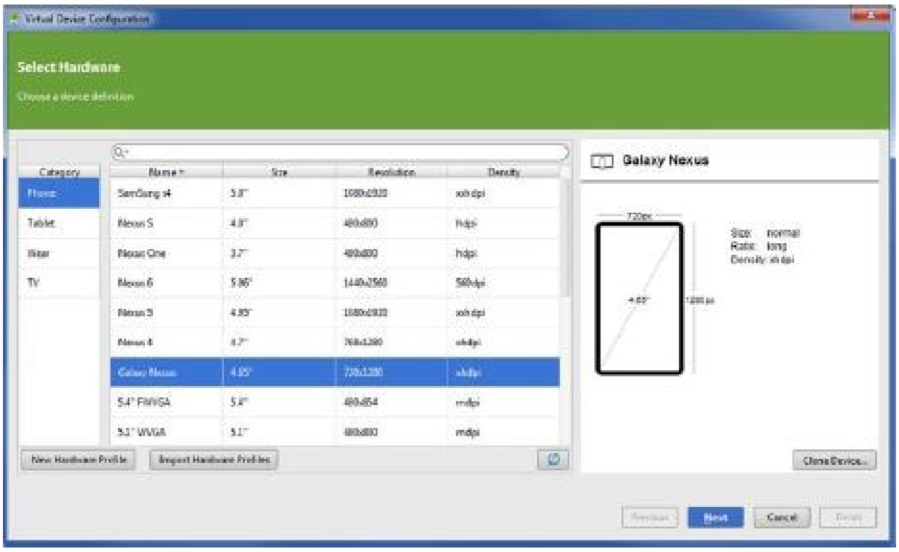


图1-26 选择Galaxy Nexus硬件

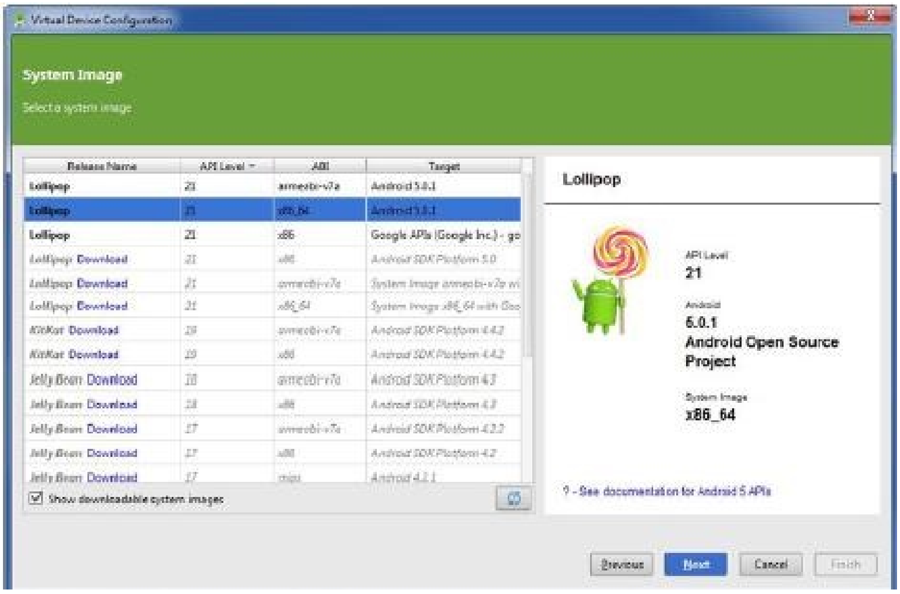


图1-27 选择 x86_64系统镜像

注意： x86_64版本需要Intel硬件加速，它仅仅工作在有限数量的Intel芯片组上。如果你安装x86_64失败，请尝试armeabi-vxx版本。

小贴士： 如果你想创建一个Android Studio还没有设备定义的设备，我们建议你去phonearena.com 查找你的型号。在那里你会找到技术规格，你可以用来创建一个新设备的定义。在你创建好了新设备的定义之后，使用相同的步骤来创建一个新的AVD。有一个优秀的第三方市场的Android模拟器叫Genymotion。Genymotion模拟器对非商业目的用户是免费的而且用起来不错。解释如何设置和使用Genymotion超出了本书的范围，但是你可以从genymotion.com上下载Genymotion模拟器。

在AVD上运行HelloWorld

在新创建的AVD中运行 `HelloWorld` 应用程序，单机工具栏上的绿色运行按钮，如图1-28.



图1-28 运行按钮

确保Launch Emulator单选按钮被选中，然后选择Nexus API 21的组合框。点击OK如图1-29所示.耐心等待几分钟启动AVD。然后你应该能够看到 `HelloWorld` 应用程序运行在你的电脑模拟器上。如图1-30所示。

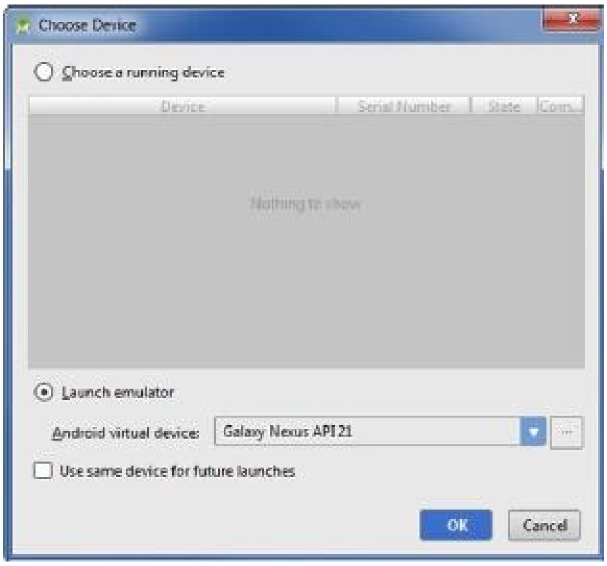


图1-29 选择设备开启模拟器



图1-30 模拟器截图

在Android设备上运行HelloWorld

如前面所说，尽管AVDs对于模拟特定的设备是有用的，尤其是那些你手头没有的设备，但是在物理设备上开发应用程序是更可取的。如果你的电脑不能通过USB数据线识别你的Android设备连接，你应该装一个USB驱动程序。如果你的电脑能够识别你的Android设备，你无需安装不同或更新版本的USB驱动程序，因为这可能导致USB连接失败。

注意： Mac和Linux用户通常不需要下载USB驱动程序来连接Android设备和电脑。

你可以在 developer.android.com/tools/extras/oem-usb.html 上找到合适的USB驱动程序，或者使用你最喜欢的搜索引擎找到适合你型号的USB驱动程序。下载驱动程序并将之安装在电脑上。在你的Android设备上，点击设置，然后打开开发人员选项。确保USB调试检查框被选中。一些设备，如三星需要密码来启用USB调试，所以你可能要使用你最喜欢的搜索引擎研究如何打开你设备上的USB调试开关。如果介绍不太清晰，YouTube上一些视频资源也可以帮助你如何开启特定设备的USB调试开关。

大多数的Android设备配备有一根一端有一个USB接口另一端有一个小型USB接口的数据线。你的Android设备通过这根数据线连接到你的电脑上。点击在图1-31中圈起来的Android 设备监听器按钮。如果驱动程序安装正确，你应该能够看到列出来的已连接设备如图1-32.

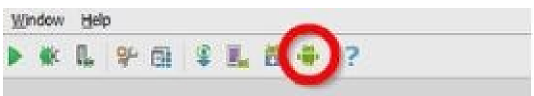


图1-31 Android设备监听器按钮

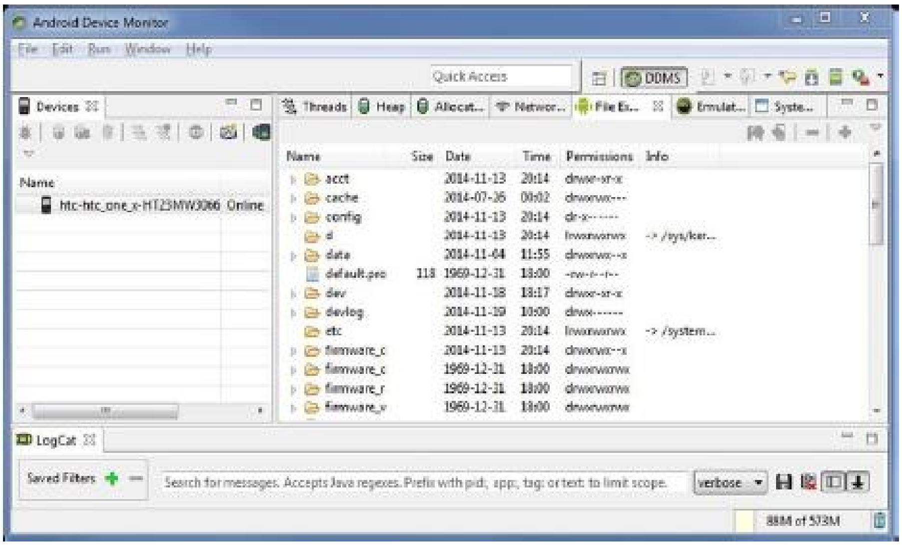


图1-32 Android设备监听器显示已连接的物理设备

注意：你的电脑和你的Android设备之间的连接是使用了一种叫做Android调试桥（ADB）的服务。如果你没有看到设备，点击IDE左下角的Terminal按钮，输入以下命令：

```
adb start-server
```

很可能在ADB服务重启之后你仍然没有看到设备。也许USB驱动程序需要电脑重启，虽然不太可能。

现在点击绿色运行按钮（如图1-28所示）。选择已连接的Android设备。在图1-33中，已连接的设备是一台HTC One X智能手机。点击OK，等待几秒钟，你将看到 `HelloWorld` 运行在你的Android设备上了。

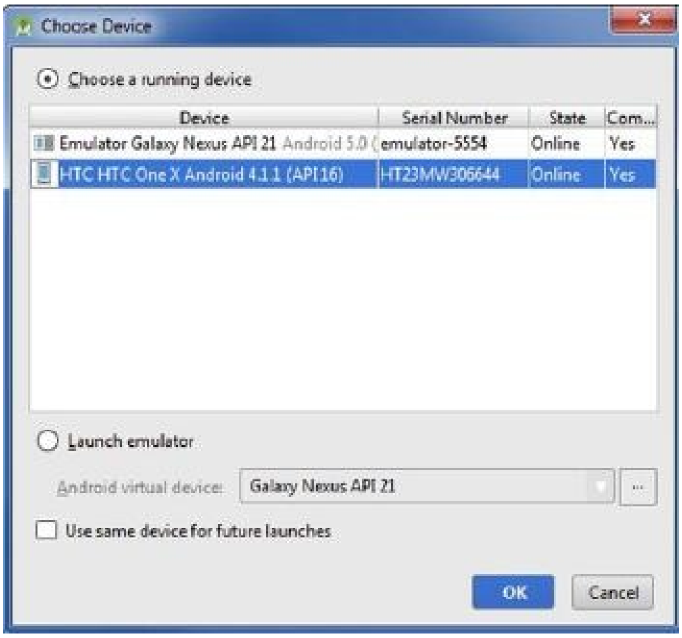


图1-33 在Android物理设备列表中选择设备

总结

在这一章，你安装了Java开发工具包，或称JDK，你也安装了Android Studio和Android SDK。你用新建项目向导创建一个简单的HelloWorld应用程序。然后你新建一个Android虚拟设备，或称AVD。我们向你介绍了如何安装所需的USB驱动程序。最后，我们向你展示了怎样在AVD和Android物理设备上运行HelloWorld。现在，你在Android Studio中拥有开发Android应用程序的所有必需的软件。

第二章：Android Studio概述

> 译者：EvilSong
>
> 来源：
>
> - Learn Android Studio 汉化教程 第二章：Android Studio概述（一）
> - Learn Android Studio 汉化教程 第二章：Android Studio概述（二）

Android Studio是一个视窗化的开发环境。为了充分利用有限的屏幕空间，不让你束手束脚，Android Studio 在特定的时间仅仅显示一小部分可用窗口。除了一些上下文敏感的窗口和上下文相关的窗口显示出来外，其他的仍旧隐藏，除非你主动打开它们。或者相反，一些可见的窗口直到你主动隐藏它们。为了充分利用Android Studio，你就需要明白这些窗口的功能以及如何、何时去显示它们。在这一章，我们将向你展示如何管理Android Studio的窗口。

任何一个集成开发环境（IDE）的基础功能就是导航。Android 项目通常是由许多的包，目录和文件组成，甚至一个稍微复杂的Android项目能够包含数百个这样的资源。你使用Android Studio的效率很大程度上取决于你处理这些资源的舒适程度。在这一章，我们还将展示在Android Studio中如何使用导航。

最后，我们将向你展示如何使用Android Studio中的帮助系统。为了充分理解这一章，我们打开第一章创建的HelloWorld项目。如果项目已经在Android Studio中打开，说明你已经准备好了。请参考图2-1，我们将讨论下面的导航功能。

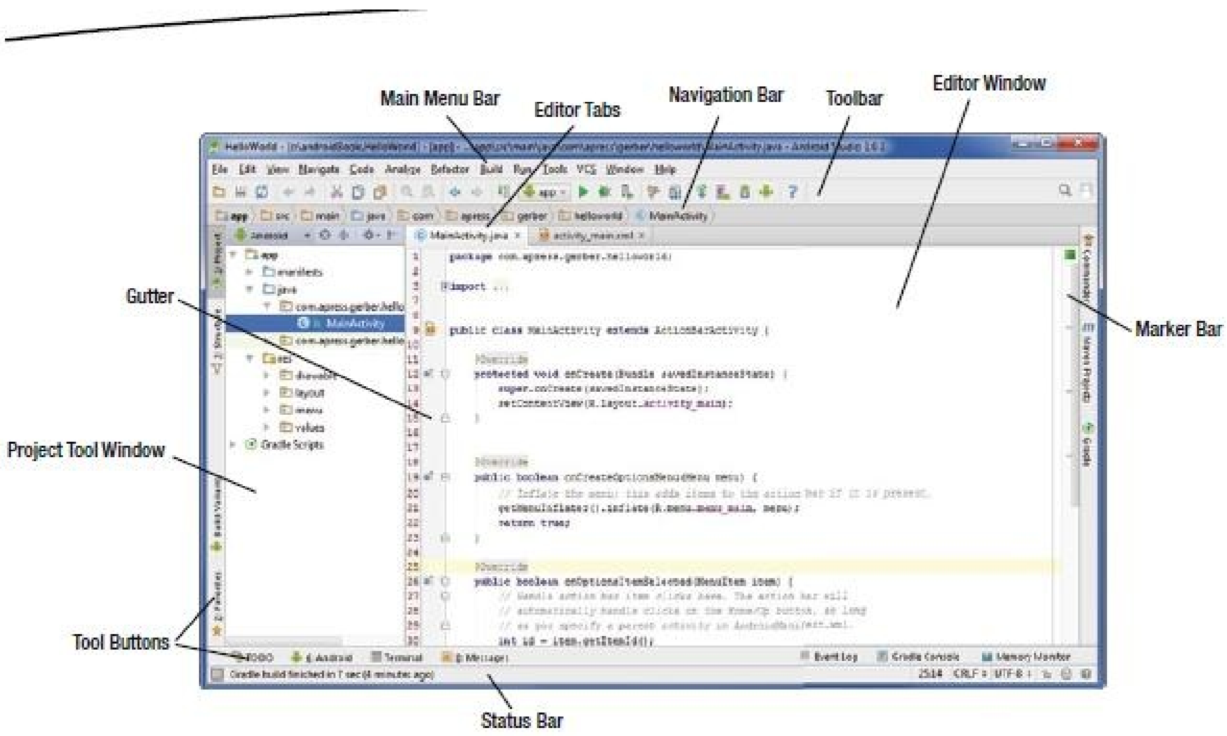


图2-1 Android Studio集成开发环境

任何IDE的主要目的就是编辑文件。正如人们所预料的，在Android Studio中允许用户编辑文件的窗口位于IDE的中心。在所有的窗口中，编辑窗口是唯一一个一直可见并位于中心的窗口。事实上，编辑窗口在Android Studio中无处不在，所以从现在起我们称它为编辑器。所有在Android Studio中围绕在编辑器周围的其他窗口都被称为工具窗口和侧窗口（左部，底部，右部）。

编辑器是一个类似于现代WEB浏览器的选项卡式的窗口。当你从工具窗口，键盘快捷键或上下文菜单打开一个文件的时候，文件将作为一个选项卡显示在编辑器中。再你新建的第一个项目—— `HelloWorld` 中， `MainActivity.java` 和 `activity_main.xml` 作为选项卡自动加载在编辑器中。Android Studio试图揣测你想开始编辑哪些文件，然后在新项目向导完成后自动作为选项卡在编辑器中打开。几乎任何文件能够在编辑器中打开，虽然原始图像和声音文件还不能在Android Studio中编辑。你也可以从工具窗口拖拽文件到编辑器，这样做可以直接作为选项卡在编辑器中打开。

编辑器上面是编辑选项卡。在左边沿是边列，在右边沿是编辑栏。让我们开始探索吧。

编辑选项卡

在Android Studio中切换选项卡可以使用 `Ctrl` 或 `Alt+左或右箭头` 。当然，你也可以用鼠标选择编辑选项卡。编辑器选项卡的选项位于主菜单栏的 `Window>Editor Tabs` 下。你在此处进行的任何操作都将应用于当前选中的选项卡上。将你的鼠标放置在 `mainActivity.java` 选项卡并右击它（Mac下按住 `Ctrl` 点击）。如图2-2所示，在生成的菜单中，你会看到很多跟 `Window>Editor Tabs` 下相同的选项。从这个菜单中，选择Tabs Placement子菜单。此菜单选项允许你将选项卡移动到上，下，左，右。将选项卡移动到左边或右边能容纳更多可见的选项卡，但是会占用部分屏幕。

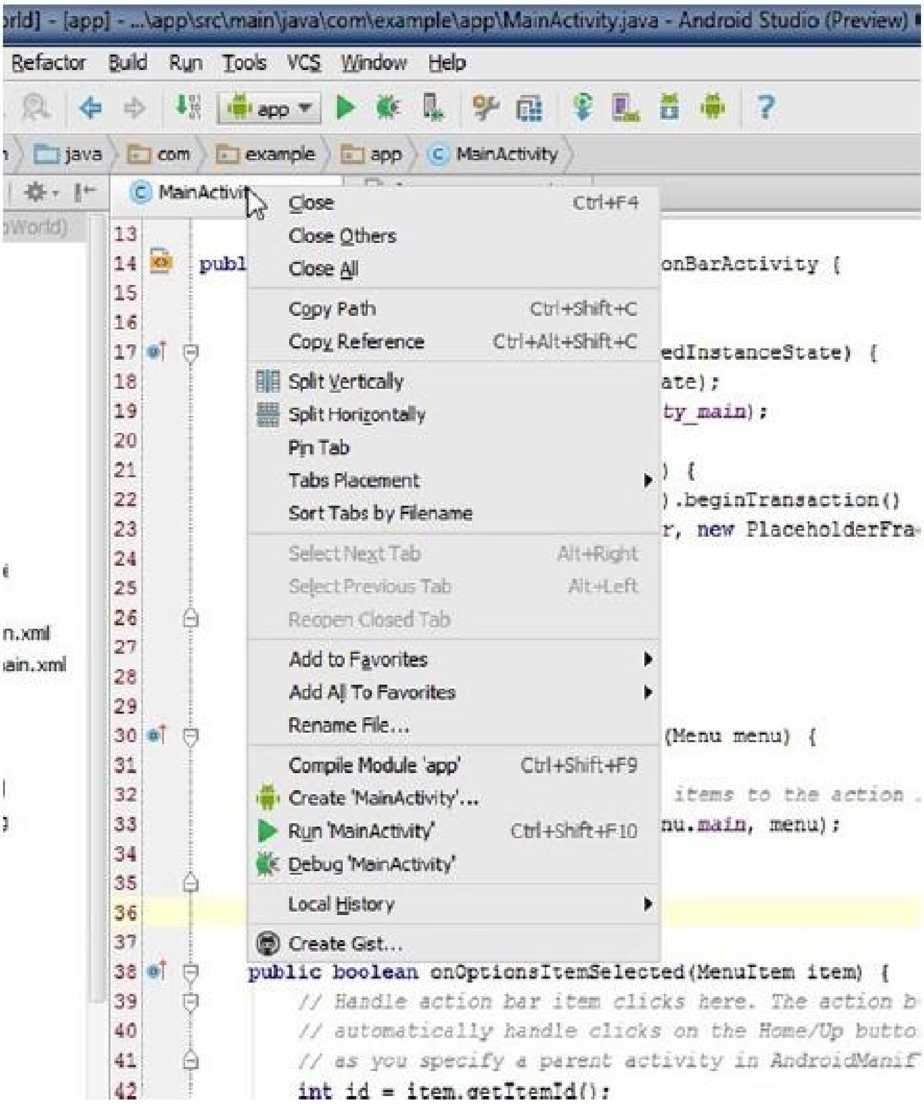


图2-2 内容菜单编辑选项卡

编辑器菜单上的Close和Close All操作是很明确的。而你想关闭除此之外的所有选项卡，你应该使用Close Others。而Split Vertically和Split Horizontally操作用来将编辑器细分成窗口。如果你想比较两个文件那么Split Vertically将会很有用。你可以无止境的进行细分，尽管这种嵌套分割的效果大大降低。你也可以将其他窗口的文件拖拽到编辑器的任何面板，或者从一个面板到另一个面板。当最后一个选项卡关闭的时候，整个面板就会消失。

边列

边列用于传递你代码的信息。也许边列最明显的特性就是在你代码的同一行用小色卡或者小图标来显示那些可视化资源。边列也可以用于设置断点，支持代码折叠，并用作代码范围指示器。所有这些特性将会在后续的章节中详细介绍。

标记栏

编辑器的右边是标记栏。标记栏用于指示你资源文件中重要行的位置。例如，当你的Java或XML文件出现警告信息或编译出错时标记栏会高亮显示这些信息。右边栏也会显示未提交的更改，搜索的结果和书签的位置。标签栏不会像边列一样滚动，相反，标签栏上的彩色标记是文件长度的相对定位。点击标签栏的彩色标记将会立即跳转到文件的位置。现在通过点击一些彩色标记来练习使用标签栏。

工具按钮

你应该已经看到了默认显示在左边面板的项目工具窗口。你可以点击主菜单上的`View>Tool Windows`来查看全部的工具窗口。现在你可以仔细的观察IDE的左边，右边和下边。你会发现工具按钮相对应的窗口。注意，有一些工具按钮后面有一个数字，这些数字结合`Alt`（Mac上的`Cmd`）键用来打开/关闭相应的工具窗口。尝试点击这些工具按钮来练习这项技能。同时使用键盘快捷键`Alt+1|Cmd+1`，`Alt+2|Cmd+2`，`Alt+3|Cmd+3`等打开/关闭工具窗口。

当某个工具窗口打开之后，相应的工具按钮呈深灰色，表示不能按下。请注意位于边缘角落的工具按钮。例如，Project的默认位置在左上角，Favorites的默认位置在左下角。

侧边栏（左边，下边，右边）可以同时被两个工具窗口共享。同时打开Favotites和Project工具窗口看看侧边栏是如何共享的。可以看到，Favorites和Project工具窗口位于同一边的相对的角落。两个位于相同角落的工具窗口是不会同时出现在一个侧窗格窗口的。例如，Project和Structure工具窗口不能够同时显示出来——至少在Android Studio的默认配置上不会。

默认布局

不要将Android Studio中的默认布局和Android SDK中的布局混为一谈。默认布局是聚集在编辑器周围的一组特定的工具窗口。Android Studio打开之后配置一个将Project工具窗口显示在左边的默认布局。以前布局显示如图2-1所示。

让我们看看主菜单上的Window菜单。前两个菜单项是Store Current Layout as Default（将当前布局存储为默认布局）和Restore Default Layout（恢复默认布局）。恢复默认布局通常用在IDE窗口变得拥挤，或者你想清理面板将之还原到你熟悉的布局。你也可用通过打开，关闭，调整或者重新定位你喜欢的窗口，然后选择将当前布局存储为默认布局来设置新的默认布局。

重新定位工具按钮

正如前面所提到的，Project和Structure工具窗口之所以不能够同时显示，是因为他们相应的工具按钮位于相同的角落。但是你可以任意移动工具按钮到你想要的角落。将Structure工具按钮拖拽到左下角的角落。现在，使用键盘快捷键`Alt+1|Cmd+1`和`Alt+7||Cmd+7`或者点击工

具按钮来打开Project和Structure工具窗口。由于我们将这两个工具按钮移动到相反的角落，所有Project和Structure能够在相同的面板窗口同时地展现出来。

导航工具窗口

本节讨论专门用作导航的工具窗口：Project，Structure，Favorites，TODO和Commander。表2-1列出了他们每个导航窗口的功能。之后的章节将讨论许多其他工具窗口。

表2-1 导航工具窗口

工具窗口	PC 快捷键	Mac快捷键	功能
Project	Alt+1	Cmd+1	允许你浏览项目中的文件和资源
Favorites	Alt+2	Cmd+2	显示收藏夹，书签和断点
Structure	Alt+7	Cmd+7	当前文件中的对象或元素以分层树的形式呈现
Commander			类似于 Project，但是允许更便捷管理文件
TODO			将项目中的所有活动事项显示在一个列表中

项目工具窗口

我们发现Project工具窗口是非常有用的导航工具窗口，因为它将有限的宽度空间和相对容易的访问结合起来。为了欣赏Project工具窗口的效用，你要将Project设置为窗口模式。Project总共有三种模式；Project，Packages和Android。Android Studio在默认情况下将模式设置为Android。Android和Project是非常有用的模式，即使Android模式可能为你隐藏某些目录。模式设置组合框毗邻IDE的左上角Project按钮并与之呈90度角。Project工具窗口提供了一个简单的嵌套接口文件和目录树来方便你切换。Project工具窗口呈现你项目中的所有包，目录和文件的概览。如果你在项目工具窗口中右击（mac下按住 `ctrl` 单击），将会显示一个上下文菜单。此处有三个重要的菜单项：复制路径，文件路径和在资源管理器中显示。点击Copy Path复制操作系统的绝对路径到剪贴板。点击File Path以倒序方式显示目录堆栈，点击任何一个目录都将在操作系统中打开他们。点击 Show in Explorer将会在你的操作系统中打开一个新窗口来显示文件。如图2-3.

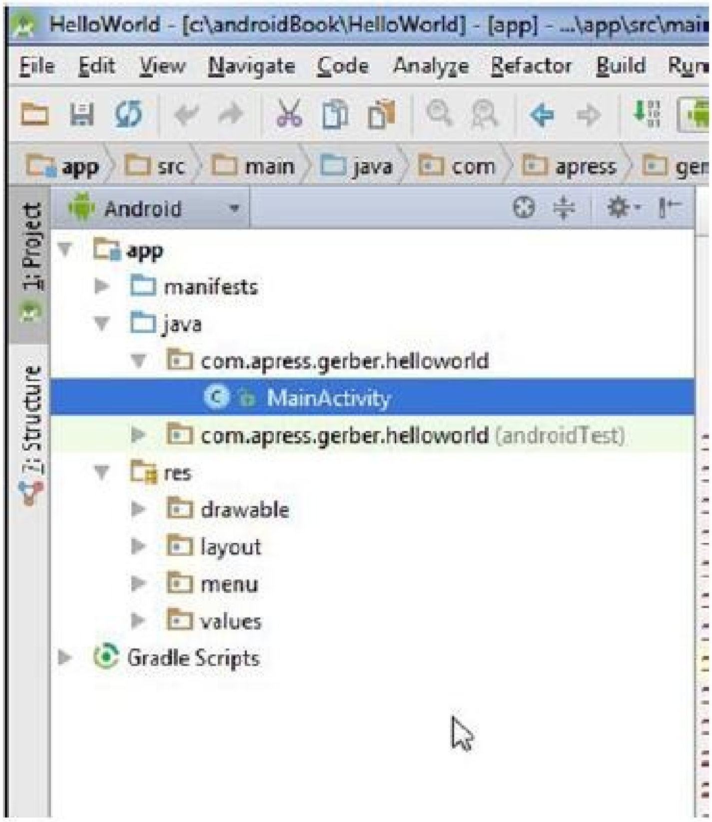


图2-3 项目工具窗口

结构工具窗口

结构工具窗口用来显示文件中元素的层次结构。当编辑器显示诸如 `MainActivity.java` 的java源文件时，Structure工具窗口将会以树状形式呈现元素，如字段，方法和内部类。当编辑器显示诸如 `activity_main.xml` 的XML文件时，Structure工具窗口以树状结构呈示XML元素。在Structure工具窗口中点击任何元素，光标将会立即移动到编辑器中的该元素上。Structure工具窗口特别适合导航超大源文件中的元素。通过打开Structure工具窗口和导航 `MainActivity.java` 与 `activity_main.xml` 中的元素来练习此技巧。如图2-4所示。

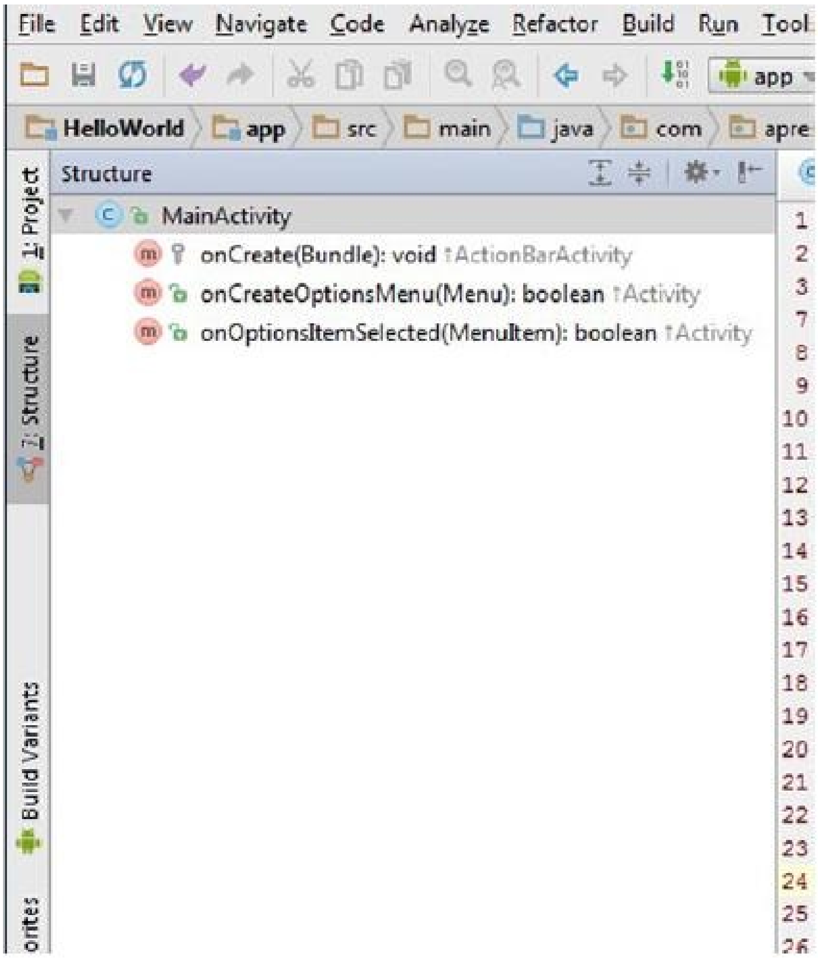


图2-4 结构工具窗口

收藏夹工具窗口

当在Android中开发一个功能（或者调试bug）的时候，你可能会创建或修改一些相关文件。中等复杂的Android项目可能包含数百单个文件，因此，具有分组相关文件的能力是非常有用的。Favorites工具窗口能够容纳那些你将 实际上位于你项目中完全不同的地方的相关文件进行逻辑分组的最爱。

将 `MainActivity.java` 和 `activity_main.xml` 文件在编辑器选项卡中打开。在任意一个选项卡上右击（Mac下按住 `Ctrl` 单击），在上下文菜单中选择Add All to Favorites。在Input new favorites list name中输入 `main` 然后点击OK。如果Favorites工具窗口没有打开，通过 `Alt+2|Cmd+2` 来打开。展开列表中的 `main` ，然后通过双击来打开/激活其中的一个文件。

就像Favorites窗口能够让你立即导航到特定文件或文件组一样，书签能够让你快速导航到文件的特定行。将你光标放到 `MainActivity.java` 的任意一行。按一下 `F11` （Mac下按 `F3` ）。这个操作便是创建或者删除资源文件包括XML文件中的书签。观察边列的检查标志和标记栏的黑色标记来指示新的标签。通过打开Favorites工具窗口中的书签来查看你刚刚创建的书签。

注意： 在PC上，如果 `F11` 没有响应，请确保你键盘上的 `F-Lock` 键是激活的。

> 译者注：Android Studio一直在升级，此处F11没有效果的童鞋可以通过查看点击菜单栏的 `Navigate ➤Bookmarks` 来查看标签快捷键。

断点是用来调试的。不像标签，可以在任何文件中进行设置，断点只能在java文件中进行设置。打开 `MainActivity.java` ，然后在如下代码行的边列上单击：

```
setContentView(R.layout.activity_main);
```

你会发现一个红色圆圈出现在边列，并且整行代码都是红色高亮的。断点只能被设置在可执行的代码行上；举个例子，如果在注释行上设置断点，并没有什么效果。打开Favorites工具窗口中的Breakpoints查看你刚刚创建的断点。你可以用断点做很多有趣的事，在之后的第12章，我们将讨论用于调试的断点。

TODO工具窗口

TODO的意思当然是要做的事。TODO本质上是用于程序员们和他们的合作者表明还要做的事。TODO写法跟注释类似，以双斜杠开头，然后TODO大写，后面跟一个空格。例如：

```
//TODO inflate the layout here.
```

在 `MainActivity.java` 中创建一个TODO然后打开TODO工具窗口，就可以看到它。在TODO工具窗口中点击一个TODO，将会立即跳转到源代码中的TODO。

Commander工具窗口

Commander工具窗口是左右窗格的导航。这些窗格的功能类似Project和Structure工具窗口。Commander窗口和其他导航窗口的不同之处在于，它在一个时间段内只显示一个目录层次，而不是显示目录树。如果你很喜欢窗口样式的导航或者你觉得Project工具窗口太繁琐，那么，Commander工具窗口可能是一个很好的导航选择。

> 译者注：本书由于出版比较早，所以Android Studio某些功能已经经过大改，比如Commander已经在新版本中消失，所以，各位童鞋可以酌情参考。

主菜单栏

主菜单栏位于Android Studio的最上面，你几乎可以利用主菜单和其子菜单来执行任何操作。不像Android Studio中其他的一些菜单，主菜单不能被隐藏。不要被主菜单和它的子菜单项吓到。即使经验丰富的Android开发者也在日常编码中仅仅使用其中的一小部分，更多的则是通过快捷键和相应的上下文菜单实现。我们将在后面的章节中讨论主菜单栏中大部分的操作。

工具栏

工具栏中包含一些频繁使用的文本操作按钮，例如剪切、复制、粘贴、撤销、重做。正如你在第一章看到的，工具栏中还包含各种各样的管理器，包括SDK管理器和Android虚拟设备管理器。工具栏中还有设置和帮助按钮以及运行和调试应用程序按钮。工具栏中所有的按钮都有相应的菜单项和快捷键。高级用户可以通过取消勾选View下的Toolbar 菜单项来释放屏幕空间。

导航栏

导航栏是以水平箭头的链状结构方式来显示从项目根目录（左边）依次到编辑器（右边）中选中的选项卡。导航栏可以用来导航你项目中的资源文件而不必通过Project或者Commander工具窗口。

状态栏

如图2-5所示（以前在图2-1），状态栏显示一些相关的和上下文敏感反馈信息，比如正在运行中的进程或者你项目中Git版本库状态的信息。现在我们将详细的讨论一下状态栏。



图2-5 状态栏

在最左边角落里的是边栏切换按钮。单击此按钮就可以隐藏或者显示边栏。另外，当你的鼠标悬停在上面的时候，会出现一个上下文菜单以便你激活任意一个工具窗口。

消息区域用来提供反馈信息，同时显示运行过程中的所有信息。当你在UI上比如菜单项或者工具栏的按钮上滚动鼠标的时候，这块区域会显示相应的提示。点击这块区域打开事件日志。

编辑器光标位置区域以行：列的形式显示编辑器中光标的位置。点击这块区域激活对话框来直接导航到你代码中特定的行。

行间隔符区域用来显示你文本文件中的回车键格式。在Windows下默认使用CRLF代表回车换行。如果是Unix和Mac机器的话则使用LF，Git中一样。如果你在Windows电脑上开发的话，Git通常会在你提交代码的时候自动将CRLF改为LF。

文本格式区域用来源文件中的文本编码格式。默认是UTF-8，是ASCII的超集，它包含了大部分的西文字母和标准Java或XML文件中的所有字符。

文件访问指示器区域能够在读/写和只读之间切换。解锁图标意味着在当前编辑器中具有读/写功能。锁定图标意味着编辑器中的文件是只读的。你可以通过点击指示图标来切换设置。

点击高亮等级按钮，激活一个带滑块按钮的对话框，在这你可以设置代码中高亮显示的等级。

默认设置是Inspections，对应一个皱着眉头的审查经理图标。此设置为了帮助你，审查经理将对你代码中的语法错误和被称作警告的可能出现的问题进行严格的检查。你可以在标记栏上看到审查经理生成的黄色警告标识。

滑动块的下一个设置是Syntax，对应一个侧着脸的审查经理的图标。这个设置下，审查经理是睁一只眼闭一只眼的。Syntax检查不如Inspections严格，但是仍然会高亮显示阻止代码编译的语法问题。

最后一个高亮模式是None，对应一个微笑着的审查经理图标。这个图标让我想到审查经理很开心并且不关心你的代码。这种模式下，即使是最严重的语法错误也会被忽略，尽管当你试图构建的时候编译器仍然会阻止。我建议将高亮等级设置为Inspections并且学会欣赏审查经理的“严厉的爱”。

常用操作

本节介绍在Android Studio中的各种常用操作。如果你使用过像微软的Word类似的文本编辑器的话 ，那么你可能对这节比较熟悉。

选择文本

正如你想的那样，任何一个好的文本编辑器，在源文件中双击任何单词就能够选中它。此外，单击并拖动光标选择单词或者字母中的文本元素。将光标放在源文件中，并按下 `Shift+下箭头` 或者 `Shift+上箭头` 来选择从光标处开始的行文本。在行文本的任何地方三击来选择整行。按下 `Ctrl+A|Cmd+A` 选择文件中的所有文本。

如果你将光标放置在任何单词上，并按下 `Ctrl+W|Alt+上箭头` 将会选中整个单词。如果你继续按下 `Ctrl+W|Alt+上箭头` ，将会无止境的选择相邻的文本。如果此时你按下 `Ctrl+Shift+W|Alt+下箭头` ，文本选择将会收缩。这种增加/减少选择的功能在Android Studio中被称为结构化选择。

使用撤销和重做

撤销和重做命令对于回滚有限数量的编辑操作是非常有用的。特殊的UI事件比如按下Enter或者重新定位鼠标用来限改变。撤销和重做的快捷键分别是 `Ctrl+Z|Cmd+Z和Ctrl+Shift+Z|Cmd+Shift+Z` 。工具栏左边的紫色左右箭头也分别代表撤销和重

做。Android Studio会记住你上次保存至今的所有步骤或者达到上限300步。撤销和重做只能在一段时间内应用到一个文件，因此，最有效方式来回滚更改是利用Git，将在第7章中讨论。

查找最近文件

Android Studio中最好的特性是它记住了你最近是用过的所有文件。选择 `View>Recent Files` 或者按下 `Ctrl+E|Cmd+E` 来激活这个命令。结果对话框允许你选择任何最近的文件，并将之作为编辑器中的一个选项卡来打开。默认只能记住之前的50个文件。你可以通过 `File ➤Settings ➤Limits ➤Editor ➤Recent Files Limit` 来修改这些限制。

遍历最近导航操作

Android Studio也能够记住你最近的导航操作。导航操作包括指针移动、标签变更和文件激活。按下 `Ctrl+Alt+左箭头|Cmd+Alt+左箭头` 或者 `Ctrl+Alt+右箭头|Cmd+Alt+右箭头` 来遍历你的导航操作历史。注意，导航操作不同于编辑操作；如果你想遍历你的编辑操作，你应该使用撤销和重做。

剪贴，复制，粘贴

如果你使用过任何的文本编辑器或者文档处理器，你应该熟悉剪切、复制和粘贴。表2-2列出了这些基本命令以及一些扩展的剪贴板命令。

表2-2 剪贴，复制，粘贴

Command	PC Keys	Mac Keys
Cut	Ctrl+X	Cmd+X
Copy	Ctrl+C	Cmd+C
Paste	Ctrl+V	Cmd+V
Extended Paste	Ctrl+Shift+V	Cmd+Shift+V
Copy Path	Ctrl+Shift+C	Cmd+Shift+C
Copy Reference	Ctrl+Alt+Shift+C	Cmd+Alt+Shift+C

除了协同剪贴板提供的一些简单的剪切、复制和粘贴操作，Android Studio扩展了剪贴板的功能，能够记住之前五步剪贴和粘贴操作。在Android Studio运行的时候，当你从Android Studio剪切或者复制文本（或者任何其他的应用程序）的时候，Android Studio将会将这些文本放到一个堆栈中。按下 `Ctrl+Shift+V|Cmd+Shift+V` 就能够看到扩展的剪贴板堆栈。结果对话框允许你选择粘贴其中一项。如图2-6所示。

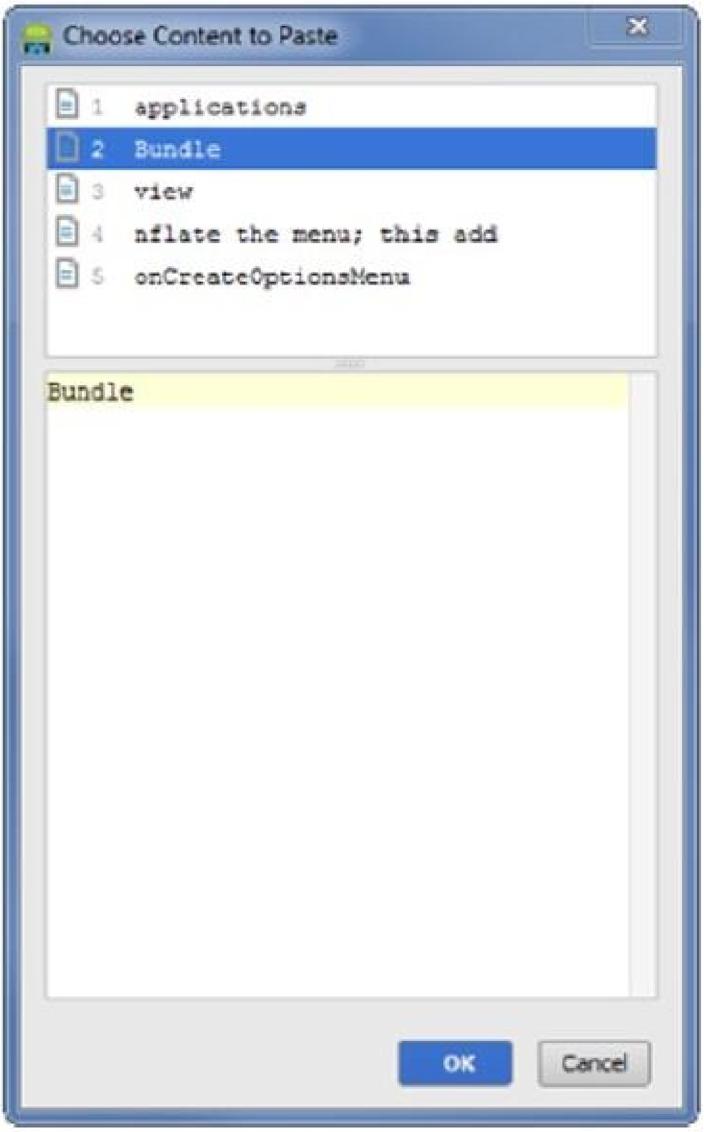


图2-6 展开剪贴板

你也可以通过导航栏上的 `File ➤Setting ➤Limits ➤Editor ➤Maximum Number of Contents` 来更改扩展剪贴板上内容的数量。你也可以通过在选中的文本上右击，然后选择Compare with Clipboard项来将当前选中的文本和剪贴板中最近的元素来进行比较。

复制路径命令 `Ctrl+Shift+C|Cmd+Shift+C` 用来复制任何文件的物理路径或者Project或Commander工具窗口中文件夹的路径或者编辑器中选项卡的路径。在终端会话中复制路径操作是非常有用的。

通过 `Ctrl+Alt+Shift+C|Cmd+Alt+Shift+C` 命令，Android Studio能够复制一个逻辑引用到方法、变量或类中。当你将这个引用复制到另一个源文件的时候，Android Studio会自动的包含所需的所有修饰符和imports。你也可以在Commander和Project工具窗口中的包、目录和文件中使用剪切、复制和粘贴来代替鼠标直接拖拽来整理你项目中的资源文件。

上下文菜单

大多数的上下文菜单都可以在IDE中通过右击来激活（mac下按住`ctrl`单击）。在前面的章节，你已经学习了编辑器选项卡的上下文菜单。Android Studio中的大多数的窗格，图标和边栏，如果你右击（mac下按住`ctrl`单击）将会出现一个上下文菜单。Android Studio的最大优势是执行一个操作仅仅一种方式。这种冗余意味着你可以根据你自己的喜好习惯来免费扩展你的技能和习惯。我发现一些频繁的操作使用快捷键，而对一些低频率的操作使用菜单和上下文菜单进行操作是非常有效的方法来使用Android Studio。接下来通过右击（mac下按住`ctrl`单击）IDE的边栏，选项卡，窗格和文件来探索上下文菜单。

获取帮助

Android Studio中的帮助菜单有几个有用的菜单项。查找操作（`Ctrl+Shift+A|Cmd+Shift+A`）是Android Studio中用的最多的获得帮助的命令。这个命令行会激活一个对话框，允许你搜索Android Studio中的任何东西。按下`Ctrl+Shift+A|Cmd+Shift+A`并且在搜索框中输入Show Line Numbers，然后使用鼠标选择Setting然后按下Enter。在设置窗口，选择`Editor ➤ Appearance`。（译者注：Android Studio是`Editor>General>Appearance`）。接着你就能看到Show Line Numbers的复选框了。

选择`Help➤Online Documentation`,你能够看到Android Studio中的所有技术规范文档资源。这是Android Studio最全面的文档。同时，`Help➤Default Keymap Reference`菜单项也是一个非常有用的参考。你可以考虑将PDF文档打印出来，然后在你学习使用Android Studio的时候放在手边时时参考。

导航快捷键

快捷键是最强大的方式来浏览Android Studio。在主菜单栏选择Navigate菜单浏览其内容。本节将讨论一些重要的菜单项（如表2-3）及其对应的导航菜单快捷键。随后的章节讨论其他的菜单项。

表2-3 导航快捷键

Command	PC Keys	Mac Keys
Select In	Alt+F1	Alt+F1
Class	Ctrl+N	Cmd+O
File	Ctrl+Shift+N	Cmd+Shift+O
Line	Ctrl+G	Cmd+L
Related File	Ctrl+Alt+Home	Alt+Cmd+Up-Arrow
Last Edit Location	Ctrl+Shift+Backspace	Cmd+Shift+Backspace
Type Hierarchy	Ctrl+H	Ctrl+H
Declaration	Ctrl+B	Cmd+B

选择在

Android Studio最好的特性之一是导航是双边的。你已经见识过如何在各种工具窗口中作为编辑器的选项卡来打开/激活文件。现在你将要学习如何从编辑器中导航到 各种工具窗口。

按下 `Alt+F1` 。激活选择上下文菜单，其中包含几个菜单项，包括项目视图，收藏和文件结构等。点击项目视图操作。项目工具窗口将会激活，文件对应的编辑器中活动的选项卡将会高亮显示，并且该文件的父目录将会自动打开。Android项目中有很多的资源文件；因此，会用选择操作是最重要的熟练技能之一。

类

类操作允许你导航到一个特定的java类。值得重点关注的是这个操作只能查找Java资源文件或是Java资源文件的内部类。按下 `Ctrl+N|Cmd+O` 然后开始输入 `act` 。Android Studio将检索你所有的文件，并且提供可能匹配的列表，将最可能匹配的高亮显示。你只需要按下 `Enter` 来打开 `MainActivity.java` 。

文件

文件操作允许你导航到项目中的任何文件。如果你想查找你项目中的xml文件，那么你将用这个操作。按下 `Ctrl+Shift+N|Cmd+Shift+O` 然后输入 `act` 。我们故意使用了相同的检索项来说明 `Navigate>File` 中的更广泛的范围。可以注意到搜索结果中包含Java资源文件 `MainActivity.java` 和一些其他的文件，如 `activity_main.xml` 。用 鼠标来选择 `activity_main.xml` 然后按下 `Enter` 键来打开。

行

使用 `Ctrl+G|Cmd+L` 来执行行操作，会激活一个对话窗口，然后以 `Line:Column` 形式导航到你的资源文件。如果你在Go to Line对话框中输入一个简单的整数然后点击OK，Android Studio将会跳转到指定的行而不管列。

相关文件

按下 `Ctrl+Alt+Home|Alt+Cmd+上箭头` 来打开相关文件夹操作是Android Studio中一个非常有用的命令行。Android项目通常有一个非常多的相关文件。例如，一个简单的Android `Activity` 通常有至少一个相对应的xml布局文件来呈现 `Activity` 的布局，和一个相对应的XML菜单文件来呈现 `Activity` 的菜单。如果你使用Fragments，这只会增加复杂性。你已经知道通过Favorites如何组织相关文件。 `Navigate ➤Related File` 中，你可以看到Android Studio中展示给你的相关文件。打开 `MainActivity.java` 选项卡，按下 `Ctrl+Alt+Home|Alt+Cmd+上箭头` 。你应该能够看到 `activity_main.xml` 列出来了。使用鼠标来选择然后按下 `Enter` 。

最后编辑位置

按下 `Ctrl+Shift+Backspace|Cmd+Shift+Backspace` 激活最后编辑位置，这个位置允许你导航到最后的编辑位置。如果你继续激活这个命令，你的鼠标将会移动到之前编辑的文件/位置，等等。

类型层级结构

Android使用的是面向对象的编程语言Java。面向对象语言的最大特点之一是继承，它促进了代码的重用和多态性。在编辑器中打开 `MainActivity.java` 文件，按下 `Ctrl+H` 来切换打开层次结构窗口。在打开的窗口你将看到一系列的层叠对象，所有的对象都可以追溯到祖先的在Java中被称作对象的所有对象。需要记住的是只有在编辑器中打开的是Java资源文件的话 `Navigate ➤Type Hierarchy` 才可以激活。

声明

声明操作允许你跳转到原始的定义方法、变量和资源。激活这个操作的另一种方式是按住 `Ctrl|Cmd` 然后在文件中的方法，变量或资源文件上滚动鼠标。如果这个元素上有下划线，你可以通过单击导航到它的声明上。在 `MainActivity.java` 中，点击方法 `setContentView` 中的任何地方然后按下 `Ctrl+B|Cmd+B` 。你将会立即跳转到该方法的声明，它位于 `MainActivity` 的超类 `ActionBarActivity.java` 中。

查找和替换

查找和替换文本是编程的一个重要部分，Android Studio有一套强大的工具来帮助你做到这一点。本节讨论一些最重要的工具。如表2-4所列。

表2-4 查找和替换

Command	PC Keys	Mac Keys
Find	Ctrl+F	Cmd+F
Find in Path	Ctrl+Shift+F	Cmd+Shift+F
Replace	Ctrl+R	Cmd+R
Replace in Path	Ctrl+Shift+R	Cmd+Shift+R

查找

查找操作用于查找在本文件中的文本。在 `MainActivity.java` 中，按下 `Ctrl+F|Cmd+F` 在编辑器的最上方弹出一个搜索栏。在搜索栏中输入 `action` ，你将注意到文件中会立即以黄色高亮显示。你也可以在标记栏上看到一些绿色的标记用来标记查找到的文本的位置。将鼠标移到查找栏的双右箭头上将会显示高级搜索选项。

路径中查找

查找路径操作比之前讲的查找操作查找的范围更广泛的范围。你也可以使用正则表达式和界定文件覆盖的结果。按下 `Ctrl+Shift+F|Cmd+Shift+F` 并且在搜索框中输入 `hello` 。默认情况下，搜索范围被设置为整个项目，但是你可以限制搜索范围到一个特殊的目录或是module。同意整个项目的默认设置并且点击查找按钮。结果将会显示在查找工具窗口。点击查找窗口中的一个条目将会立即将关闭的文件打开为编辑器中一个新的选项卡并且跳转到搜索的地方上。

替换

通过 `Ctrl+R|Cmd+R` 打开替换操作，这个操作用于替换当前文件中的文本，并且替换功能是查找的超集。替换文本最安全的方法就是使用 `Refactor>Rename` 命令，稍后我们将介绍它。

路径中替换

通过 `Ctrl+Shift+R|Cmd+Shift+R` 打开在路径中替换操作，这个操作是在路径中查找的超集。然而，使用 `Refator>Rename` 会比在路径中替换更好，因此，当你引入错误的时候你将及其谨慎的使用这个命令。

总结

在本章中，我们讨论了编辑器和编辑器周围的工具窗口。我们讨论了如何使用工具按钮和将他们重定位。我们也讨论了包括主菜单栏，工具栏，状态栏，边栏和标记栏的用于导航的工具窗口和主要的UI元素。我们也讨论了如何去搜索和通过使用菜单栏和快捷键来导航，以及使用查找和替换。最后，我们讨论了Android Studio中怎样使用帮助系统。更重要的是，我们已经在Android Studio中建立了一个UI元素，这些我们将在后续章节中引用。

第三章：使用 Android Studio 编程

> 译者：keyoo
>
> 来源：Learn Android Studio 汉化教程 第三章：使用 Android Studio 编程

本章包含如何在Android Studio中书写或生成代码。 Android Studio 使用面向对象编程的思想来生成极其相关的和格式规范的代码。本章的特色体现在重载方法，Java块的包围语句，使用模板插入代码，自动补全，注释代码和移动代码，如果你读这本书的目标是要精通Android Studio，那么你需要密切关注这章，因为所描述的工具和技术将在你的编程效率发挥最大的作用。 让我们开始吧，若是在第一章创建的 `HelloWorld` 应用程序还没有打开，请前往打开。

代码折叠是一种在编辑器中保存屏幕实际使用空间的方式，代码折叠允许你去隐藏特定代码块以便于于你可以充分地集中注意力在你感兴趣的代码块上，如果 `MainActivity.java` 没有打开，按 `Ctrl+N | Cmd+O` 并输入 `Main` ，按 `Enter` 键打开 `MainActivity.java` 的类，如图3-1

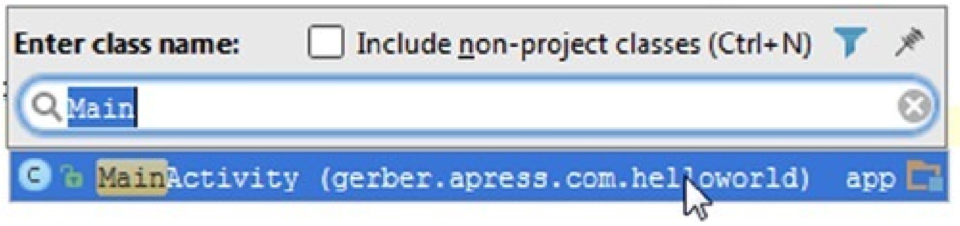


图3-1 使用输入类名对话框来打开MainActivity.java

如果行号默认没有显示，导航到 `Help ➤ Find` ，输入 `show line numbers` 并且勾选Show Line Numbers Active Editor这一项，如图3-2.

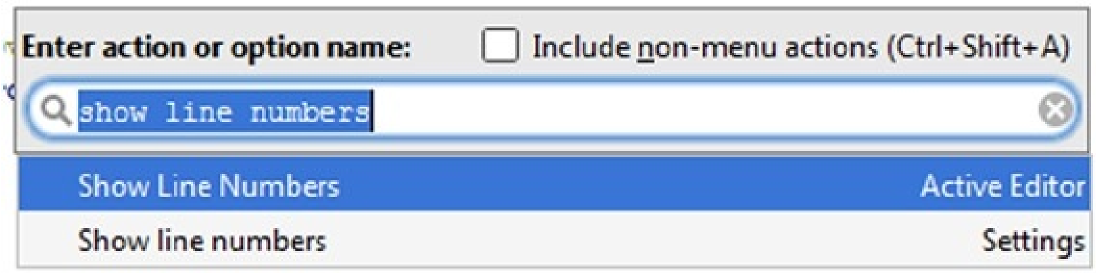


图3-2 使用输入Action或者选项名对话框来显示行号

正如你在 `MainActivity.java` 中所观察到的行号，你将会注意到一些奇怪的情况：行号不是连续的，在图3-3中，行号从1,2,3开始，然后就跳跃到7,8,9了，仔细观察图3-3的第3行，你会发现这里在 `import` 语句的左边有一个包含了 `+` 的方块，同时随后是省略号，如果你仔细观察你的代码，你也会注意到这个省略号用浅绿色高亮显示，这些可见的元素提示你Android Studio 正隐藏被折叠的代码块

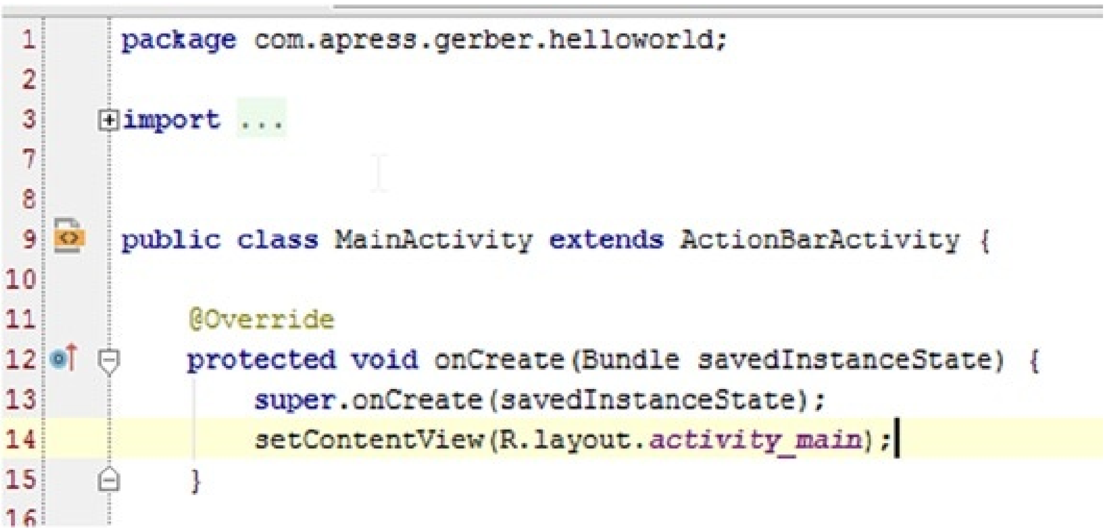


图3-3. 在import 语句的折叠代码块

位于左边留空处是折叠边框形成的细虚线，同时介于灰色留空与白色编辑器之间，折叠边框包含了三个图标:位于小方块的内部 `+` 号（正如图3-3的第3行）和有着水平线在里面的上下箭头，向下的将头表名是一个可折叠代码块的开始部分，同时向上箭头指示它的结束部分，上文提到的 `+` 号，表明某个代码块已经被折叠了，点击这些图标中的任意一个将会切换对应代码块为折叠还是打开状态，表3-1包含了代码折叠操作的描述和快捷键。

Table 3-1. Code-Folding Options

Option	PC Keys	Mac Keys	Description
Expand	Ctrl+Numeric-Plus	Cmd+Numeric-Plus	Expands the collapsed block where your cursor is located
Collapse	Ctrl+Numeric-Minus	Cmd+Numeric-Minus	Collapses the expanded block where your cursor is located
Expand All	Ctrl+Shift+Numeric-Plus	Cmd+Shift+Numeric-Plus	Expands all code in a window
Collapse All	Ctrl+Shift+Numeric-Minus	Cmd+Shift+Numeric-Minus	Collapses all the code in a window
Toggle Fold	Ctrl+Period	Cmd+Period	Collapses/expands the block where your cursor is located

把你的光标放在 `MainActivity.java` 的 `onCreate()` 方法中，现在长按 `Ctrl+Period | Cmd+Period` 来切换他的代码块展开或折叠，此外，也可以用快捷键 `Ctrl+数字键"+"` 或者 `Cmd+数字键"+"` 展开代码、也可以用快捷键 `Ctrl+数字键"-"` 或者 `Cmd+数字键"-"` 折叠代码。

总之通过点击在折叠边框内的代码折叠图标折叠/展开你的代码块，记住，折叠单个块，多个块，或一个文件的所有块，是为了节省屏幕空间从你的视线中删除了，然而当你bulid的时候，编译器仍然会编译他们。类似的，折叠一个包含了问题的、有错误的代码的块将不会从标示条删除任何警告或者出错。你若要改变代码折叠的选项，请选择通过菜单选线进入 `Settings ➤ Editor ➤ Code Folding`.

使用代码补全

现代大多数编译器都会提供一些代码补全的格式，Android Studio也不例外，Android Studio乐意帮助，就算你不主动去寻求帮助，实际上，Android Studio将为你各种各样的的默认提醒来随着你的输入完成你的代码。这些Android Studio自动生成的默认提醒列表可能是不完美的，但是这些提醒是根据最佳实践得到来设置的，并且它们通常遵循很好的命名规则。Android Studio可以很好地理解Android SDK和java程序设计语言；事实上，它可能知道那些问题，远远超过你所知道的，如果你以谨慎和热忱的心态学习这个工具的使用，你最终将会看起来想一个摇滚歌手，不管你的之前的编程经验是什么。 这个代码补全的特点是对上下文敏感的，在此条件下，这些补全根据你的光标所在的范围来提供给你不同的提醒，比如，你正在键入一个类范围内的代码，代码补全的提醒将是不同于你在方法范围内键入的代码，即使你选择不接受这个代码补全提醒，基于上述理由，你也需要注意他们。

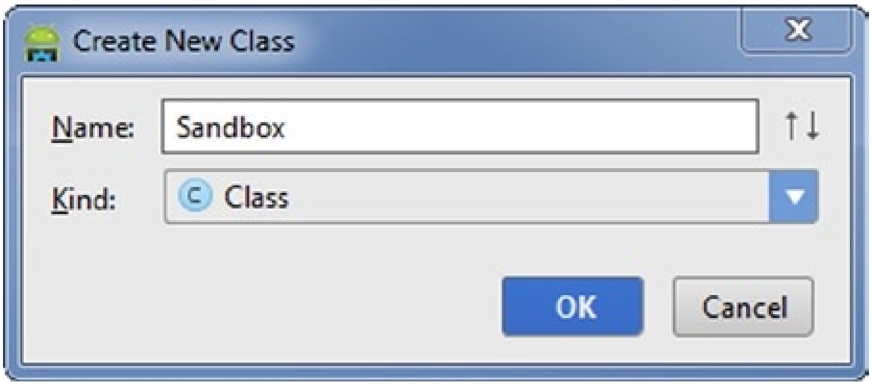
表3-2列出了4种在Android Studio中的代码补全：

- 一旦你键入代码，默认的代码补全功能会自动启作用
- 基本的代码补全不仅扮演着类似于默认代码补全的功能，还在提醒列表中的当前选项的旁边展示了一个Javadoc的窗口
- 精妙的代码补全同时扮演者Javadoc的功能之外，还有生成一个更敏感的相关的提醒列表
- 循环地扩展单词环，允许你选择在已用过的源代码单词

Table 3-2. Code-Completion Options

Option	PC Keys	Mac Keys	Description
Default	None	None	Default code-completion behavior. Android Studio displays a suggestion list next to your cursor as you type. You can use your up- and down-arrow keys to navigate among entries in the suggestion list, and the Enter key to select an entry.
Basic	Ctrl+Space	Ctrl+Space	Basic code completion functions like Default code completion, but also displays the Javadoc window next to the currently selected entry. Clicking the up-arrow icon in the Javadoc window displays detailed documentation.
SmartType	Ctrl+Shift+Space	Ctrl+Shift+Space	SmartType code completion functions like Basic, but generates a more selective and relevant suggestion list.
Cyclic Expand Word	Alt+/	Alt+/	Offers words already used in your document. Cycles up.
Cyclic Expand Word (Backward)	Alt+Shift+/	Alt+Shift+?	Offers words already used in your document. Cycles down.

让我们开始编码来演示代码补全是怎么工作的，在包 `com.apress.gerber.helloworld` 右键(Mac中是 `Ctrl` -单击)，并选择 `New ▸ Java Class` 来创建新的类对话框，如图3-4所示，吧这个类命名 `Sandbox` 并点击OK



创建新的类对话框

`Sandox.java的Sandbox` 类的大括号内部，定义一个私有变量，如图3-5，一个代码补全的菜单出现了，并列出了可能的选择帮你完成你的嗲吗，使用上下箭头键来导航代码补全的菜单，使用下键选择 List<E> 这个选项并回车。

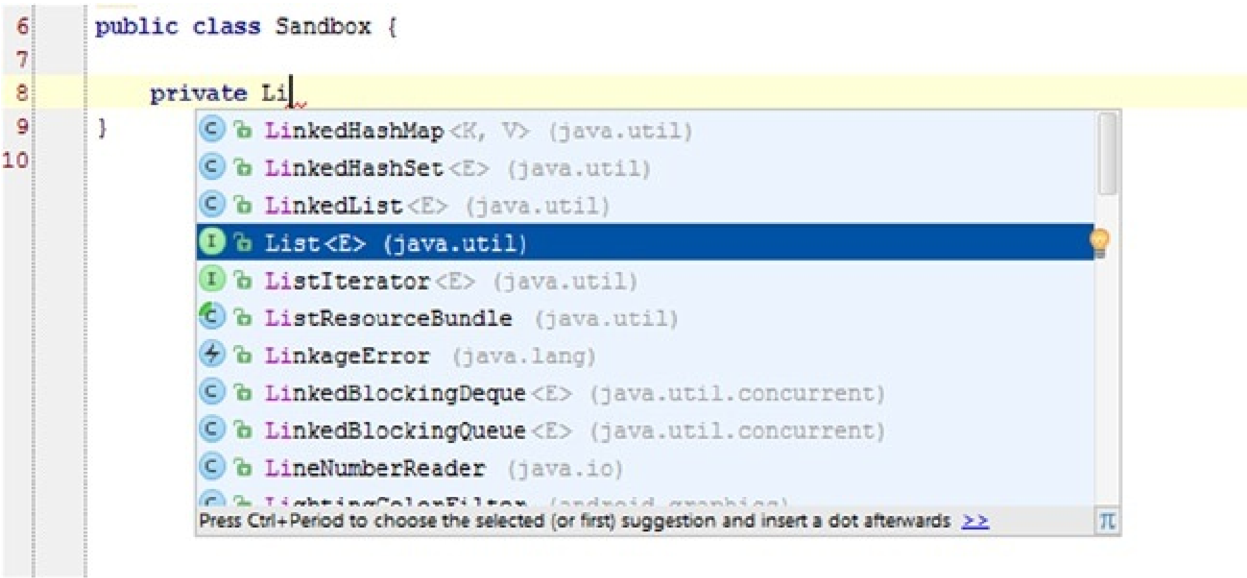


图3-5. 当你开始输入的时候出现的一个代码补全的菜单

Android Studio的默认行为是当你开始键入时，出现代码补全提醒列表，你不需要激活任何快捷键来调用这个默认的代码补全——他是自动出现的。你应该现在有一行私有 List的代码，如图3-6，直接跟随这个单词列表，键入左尖括号用以定义在java中的泛型，注意Android Studio关闭了括号语句用封闭的右尖括号，并把你的光标置于括号中

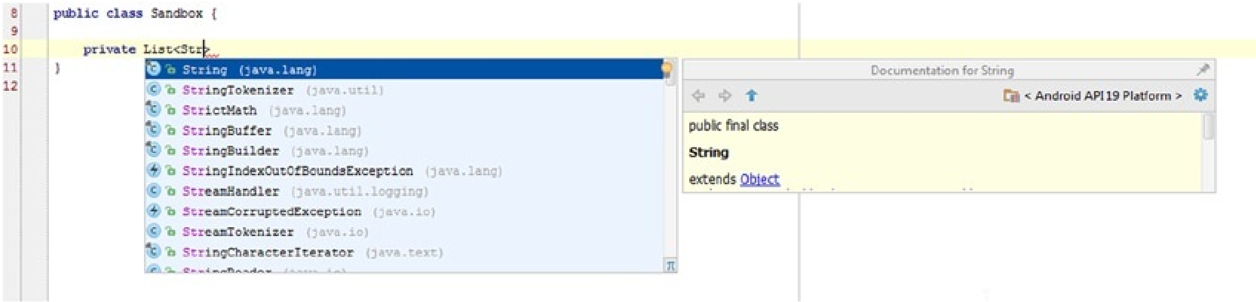


图3-6. 字符串作为泛型的列表的代码补全

在尖括号的 `Str` 类型，按下 `Ctrl+Space` 调用基本代码补全功能。你将会注意到一个 `String` 的Javadoc窗口的文档在提醒列表中出现当前选中项(`String`)，滚动 Javadoc 窗口可以看到对 `String` 的Javadoc文档说明，点击向上箭头，Javadoc窗口将展示出详细的 `String` API文档在你的默认浏览器，返回 Android Studio 并选择 `String` 作为泛型类，你将会使用的，按 `Enter` 键定义出 `List<String>` 。

Android Studio的特色之一是它会为你提醒变量名，直接键入一个空格后，私有 `List<String>` 通过按Ctr+空格出现基本的基本代码补全，Android Studio生成一个提示列表，但是上述变量名都不足于描述，因此只能键入 `mGreetings` 。小写字母 `m` 代表成员（即域），把类的成员名用 `m` 作为前缀在Android中是一种命名约定。同样的，静态类成员用小写字母 `s` 作为前缀，你不需要遵循这种命名约定，但是你的代码会更容易被别人阅读如果你遵循了，记住局部变量（方法作用域内）不允许用 `m` 和 `s` 作为前缀约定，修改你的代码行为 `List<String> mGreetings = new` 。按 `Ctrl+Shift+Space` 调用智能代码补全，选择 `ArrayList<>()` 完成语句，还包含了终止的分号，如图3-7，类似于基本代码补全，在生成提醒列表的时候，还比默认的和基本代码补全范围增多了，比如，当使用智能代码补全在赋值操作的右边，提醒列表通常包含了相关的工厂方法。

注：如果你的在 Android Studio 的JDK版本是7或者更高，通过代码补全的生成的代码可能是用菱形标记，比如，在使用泛型的赋值语句来定义的右边， `ArrayList<String>` 可能出现为 `ArrayList<>` 。

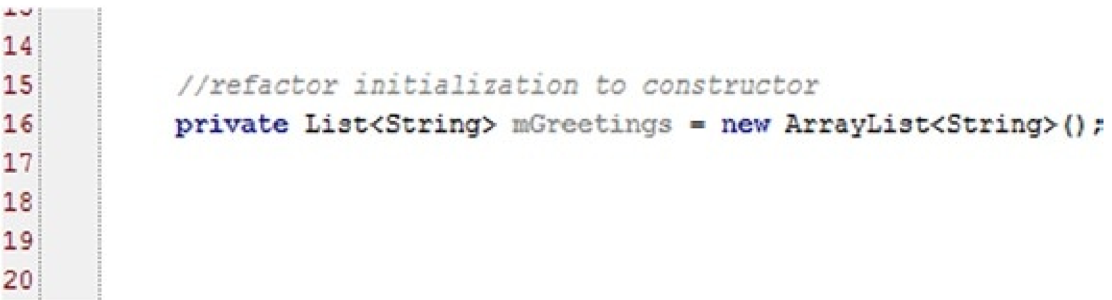


图3-7 智能代码补全

循环扩展词是一个花哨的名字，但是它实际上非常简单。调用循环扩展词只需按下 `Alt` 键，同时按住 `/` 键几秒钟，单词集合里提供你同样的单词，他们都曾出现在你的文档里，由于你的循环遍历这些单词，注意黄色高亮部分，现在调用循环扩展词，长按 `Alt + Shift +"/"` ，注意到提供并高亮的单词现在向下远离你的光标，而非向上远离。

注释代码

如果你已经完成了所有的编码工作，你了解到，有些代码行是一些注释，将会被编译器忽略，但是包含了信息或者元数据，数据是非常重要的，对于编码者和协同编码者，注释可能是行注释，以 `//` 开始，也可能是块注释，以 `/` 开始，以 `/` 结束，从主菜单可以看出，你可以激活注释，只需选择 `Code ➤ Comment` ，然而，最好的方法是通过快捷键来截获，如表3-3所示：

Table 3-3. Commenting Options

Option	PC Keys	Mac Keys	Description
Toggle Comment Line	Ctrl+/	Cmd+/	Toggles a line between comment or uncomment using Java line comment style (e.g., // ...) You may apply this action to more than one line by selecting those lines.
Toggle Comment Block	Ctrl+Shift+/	Alt+Cmd+/	Toggles selected text between commented block or uncommented block using Java block comment style, such as /* ... */. Applying a comment block to selected text will include all the selected text in the comment block.

类型重构初始化被构造，并放在 `mGreetings` 声明的上面， `Ctrl+/ | Cmd+/` 把其转化为注释，如图3-7，在开启和关闭注释切换实验，只需使用快捷键 `Ctrl+/ | Cmd+/` .

使用代码生成

恰当的使用代码生成功能，这一特色将为你节约大量的时间，代码生成是生成各种方法的的强大功能，包括了构造， `getters` , `setters` , `equals()` ， `hashCode()` , `toString()` ,方法等等。在你使用代码生成之前，确认Android Studio 是配置好了，可以忽略成员名称的前缀，如m和s，点击 `File ➤ Settings ➤ Code Style ➤ Java ➤ Code Generation` 将得到设置对话框，将会出现代码生成的标签页，如果域和静态域文本框不包含 `m` 和 `s` ，则键入他们并点击”应用“和”确定“，如图3-8.

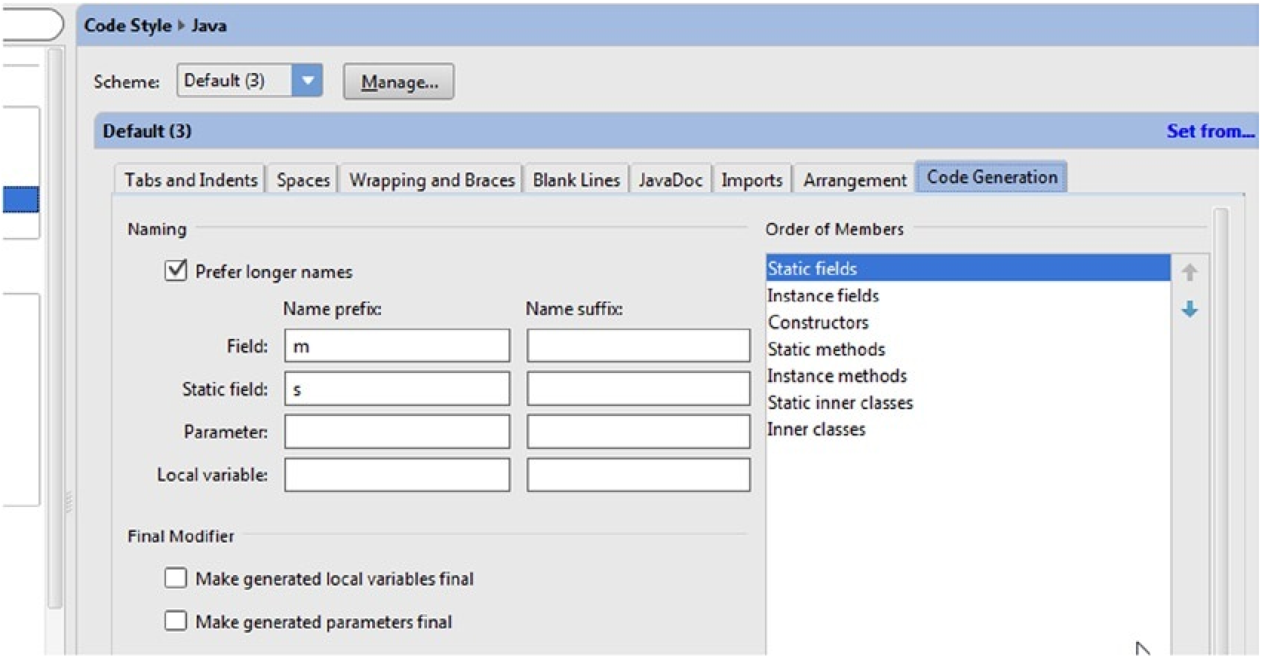
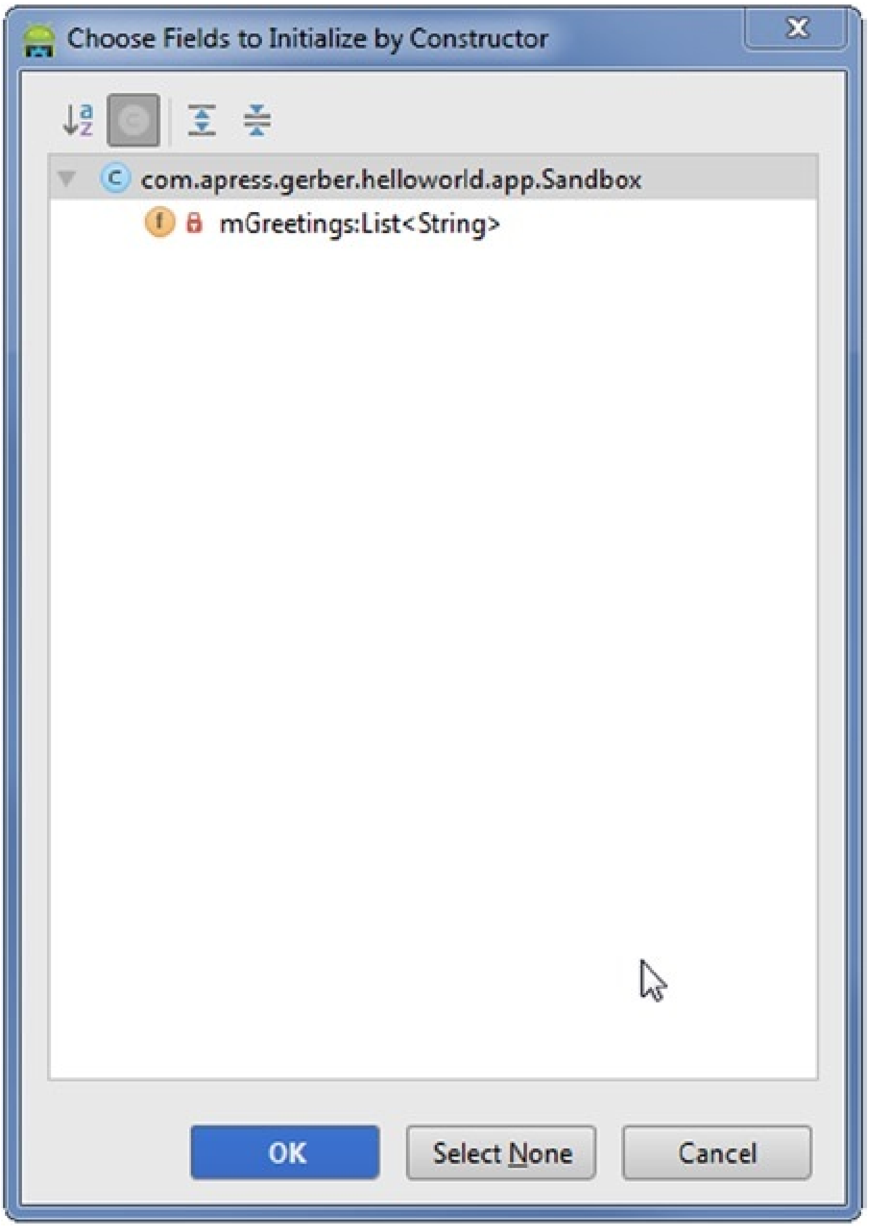


图3-8 加入 `m` 和 `s` 到代码生成标签页的域和静态域

构造器

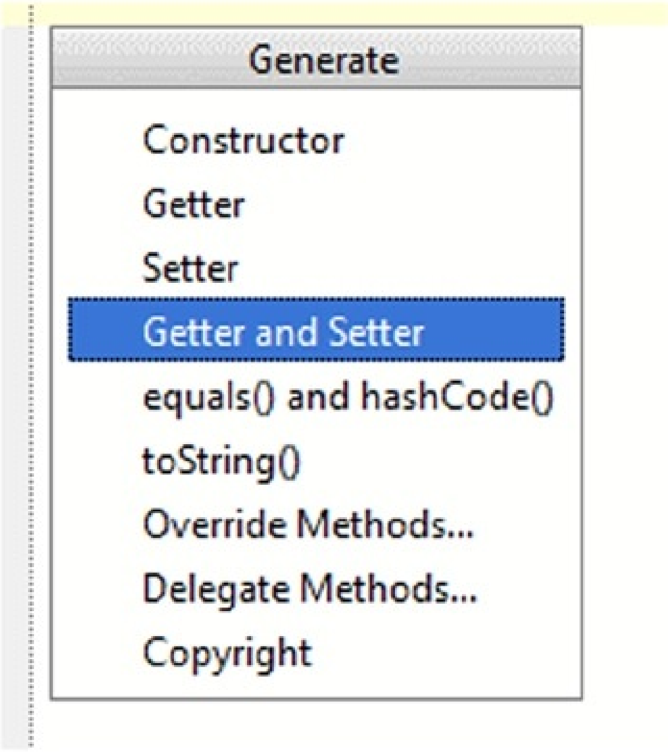
把你的光标放在 `Sandbox.java` 的类范围中。为了在Android Studio生成一个构造器，按 `Alt+Insert | Cmd+N` 并选择 `Constructor` ，那么选择的域被构造对话框初始化，如图3-9，选择类成员作为参数。我们非参数构造器，因此点击None这个按钮。在Java中，是很常见的，就是重载构造器，通过不同类型的参数和不同数量的参数，比如，你可以调用这个对话框来生成一个带 `List<String>` 参数的构造器，并且为成员变量 `mGreetings` 赋值。



通过构造对话框来选择要初始化的域。

Getters/Setters

Java类通常是封装好的,这意味着类成员通常声明为私有的,并且访问这些成员的公共接口是通过公有访问器(`getter`)和公有修改器(`setter`)方法爱提供的。在 `Sandbox.java` 类域范围内点击你的鼠标。并且按 `Alt+insert` （ `Cmd + N` ）。你会注意到有一个选项是 `Getter` ,一个为 `Setter` ，还有 `Getter` 和 `Setter` 的组合。 `Getter` 和 `setter` 方法一般都是成对出现的,除非你有一个很好的理由要省略其中一个或着另一个,否则最好是生成后面的组合选项。从列表中选择Getter and Setter, 如图3-10所示。在随后的选择字段生成 `getter` 和 `setter` 对话框中,从列表中选择 `mGreetings:List<String>` 列表并点击OK。你的类现在为变量 `mGreeting` 有了 `getter` 和 `setter` ,如图3-11。请注意,生成的代码忽略了 `m` 前缀，在生成方法名称的时候,因为你在之前的设置中把 `m` 和 `s` 设置为了前缀。



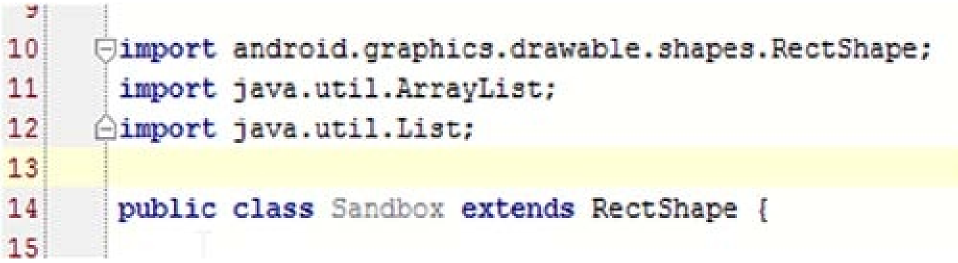
生成访问器和修改器

```
10  public class Sandbox {
11
12
13      //refactor initialization to constructor
14      private List<String> mGreetings = new ArrayList<String>();
15
16
17      public Sandbox() {
18
19      }
20
21      public List<String> getGreetings() {
22          return mGreetings;
23      }
24
25      public void setGreetings(List<String> greetings) {
26          mGreetings = greetings;
27      }
28
29  }
```

生成访问器和修改器方法

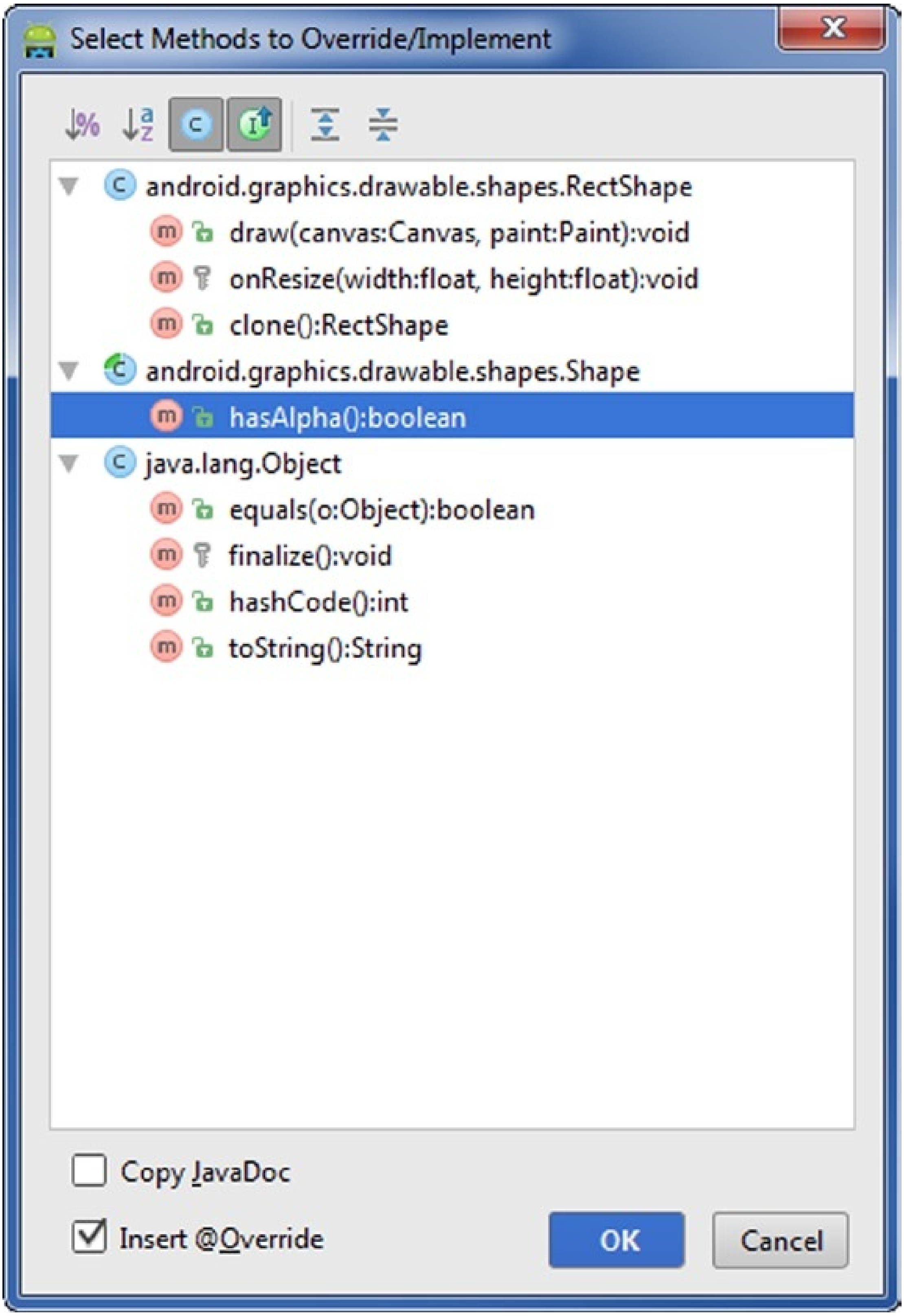
方法重载

代码生成是理解类继承结构的,所以你可以覆盖包含在任何父类或接口实现的方法。 `Sandbox.java` 是一个简单的普通java对象(POJO)。现在你修改 `Sandbox` 类来继承 `RectShape` 。当你输入继承 `RectShape` 时,这个词 `RectShape` 可能以红色突出显示。如果适合你需要的情景,按 `Alt + Enter` 来导入 `RectShape` 类,如图3-12所示

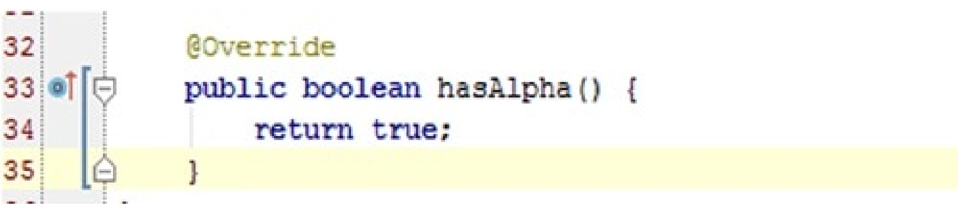


继承超类

如果你通过按 `Ctrl+H` 调用了层次结构视图,您将看到带有 `RectShape` 的 `Sandbox` 的类层次结构, `Shape` ,和 `Object` 作为它的祖先,可以通过观察图3-13，你可以明白。现在按 `Alt +Insert | Cmd + N` 并选择覆盖方法。让我们重写 `Shape` 的 `hasAlpha()` 方法,如图3 - 14所示。自java 5 版本开始的约定：要用 `@Override` 来注解想覆盖的方法，我们选中 `@Override` 复选框并插入。 `@Override` 注解告诉编译器来验证方法的名字和签名,以确保该方法事实上已被覆盖。修改 `hasAlpha()` 的返回值为 `true` 。



在RectShape中选择要覆盖/实现的方法



修改 `hasAlpha( )` 方法

`toString( )` 方法

Android Studio 可以为你生成 `toString()` 方法。让我们为 `Sandbox` 创建一个 `toString()` 方法，同时也包括 `mGreetings` 成员。按 `Alt +Insert| Cmd + N` 并选择 `toString()` 。选择你的唯一成员—— `mGreetings` ,并单击OK。Android Studio生成一个返回字符串，如 `“Sandbox{“+”mGreetings = " + mGreetings +‘}’` ,如图3-15。如果在类中有多个成员，可选择他们，他们也会被附加到这个方法的返回字符串。当然 `toString()` 方法生成的代码并不是一成不变的，这个方法是可以改变为任何你想要的，只要它返回是一个字符串。

```
37      @Override
38      public String toString() {
39          return "Sandbox{" + "mGreetings=" + mGreetings + '}';
40      }
```

图3-15 生成 `toString()` 方法

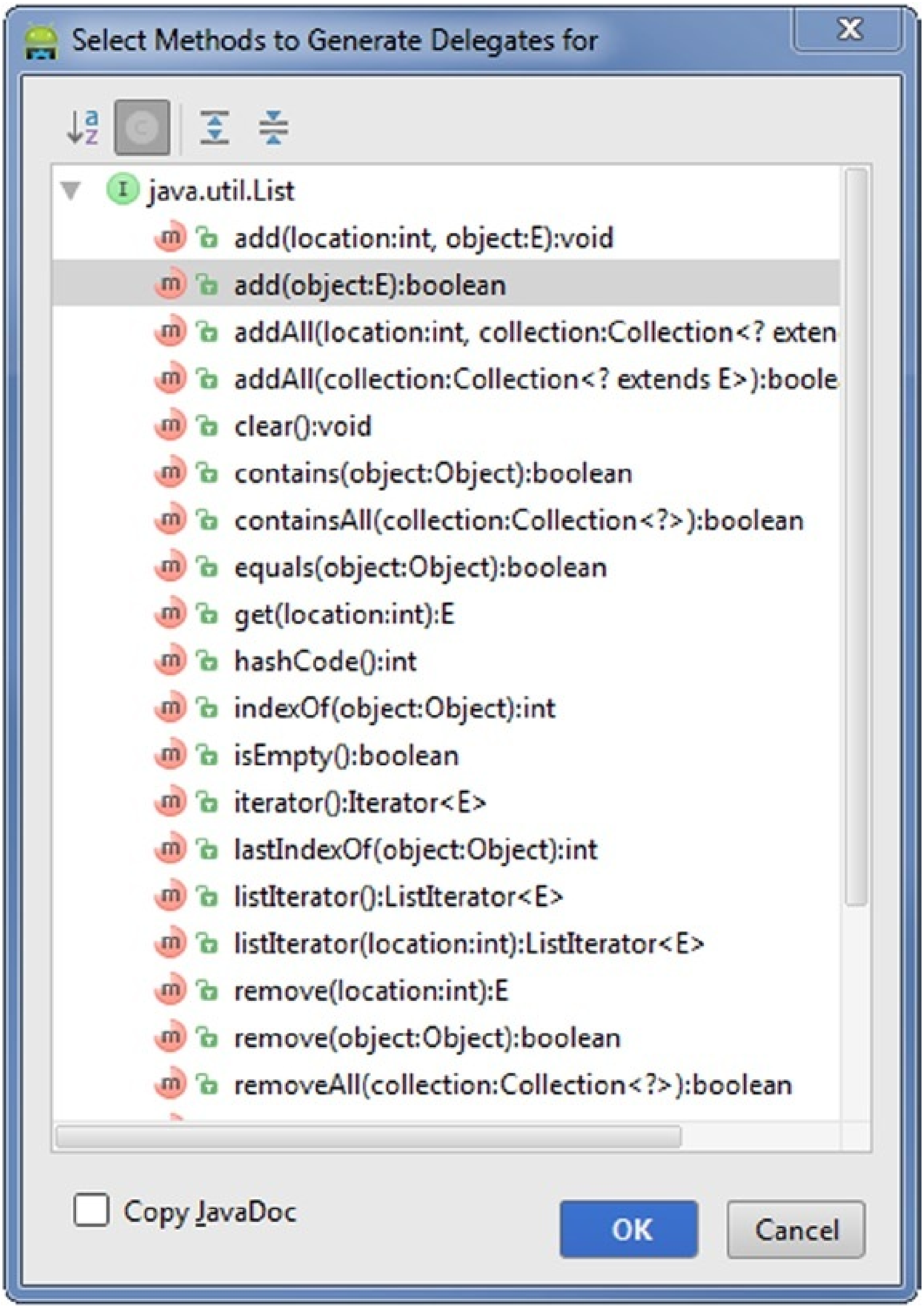
委托方法

Android Studio知道你的类成员，从而允许您从一个代理委托行为方法中定义类方法的类成员。这听起来很复杂，但它很容易。为了给你展示委托方法的选择是如何工作的，让我们转到代码中来看。在 `Sandbox.java` ，把你的光标放在类范围内。按 `Alt +Insert | Cmd + N` ,然后选择Delegate Methods。选择 `mGreetings:List<String>` 并按下OK。列表接口有很多方法，你都可以委托行为。为简单起见，选择添加 `(object:E):boolean` ,如图3-16。如果你想委托多个方法，在选择这些方法的时候，按住 `Ctrl` 键(`Cmd` 键在Mac)，单击OK。

```
43      public boolean add(String object) {
44          return mGreetings.add(object);
45      }
46
```

选择要生成委托的方法

在 `Sandbox.java` 中生成的 `add()` 方法现在是一个代理，代表 `mGreetings` 成员行为的 `add()` 方法。如图3-17所示。注意到 `add()` 方法的参数是定义为一个 `String` 来匹配 `mGreetings` 的泛型定义， `List<String>` 。委托方法并不覆盖方法，所以您可以随意重命名你的代理方法，但这个名字 `add()` 很有意义，所以尽可能保留这个名字。



生成 `add( )` 方法

插入动态模板

Android Studio有很多模板,允许您将预定义的代码直接插入到你的源文件中。在许多ide,生成的代码只是从模板中粘贴，而从来不考虑作用域;但是Android Studion的模板是对作用域敏感的,也可以集成变量数据。在你开始使用Android Studion的动态模板之前,让我们探索动态模板

和自定义模板。导航到 `File ➤ Settings ➤ Live Templates` 。选择普通模板组。现在在右上角单击绿色加号按钮并选择住模板。如图3-18，填充好缩写、描述和模板文本字段。在这个模板可以应用之前,您必须单击Define按钮,这看起来像一个蓝色的超文本链接，位于窗口的底部。现在选择Java和选择所有范围(语句,表达式,声明等等)。单击Apply。

```
23
24      //###############################################
25      // CONSTRUCTORS
26      //###############################################
27      public Sandbox() {
28
29      }
30
```

图3-18 创建一个名为 `cb` 的动态模板(注释块)

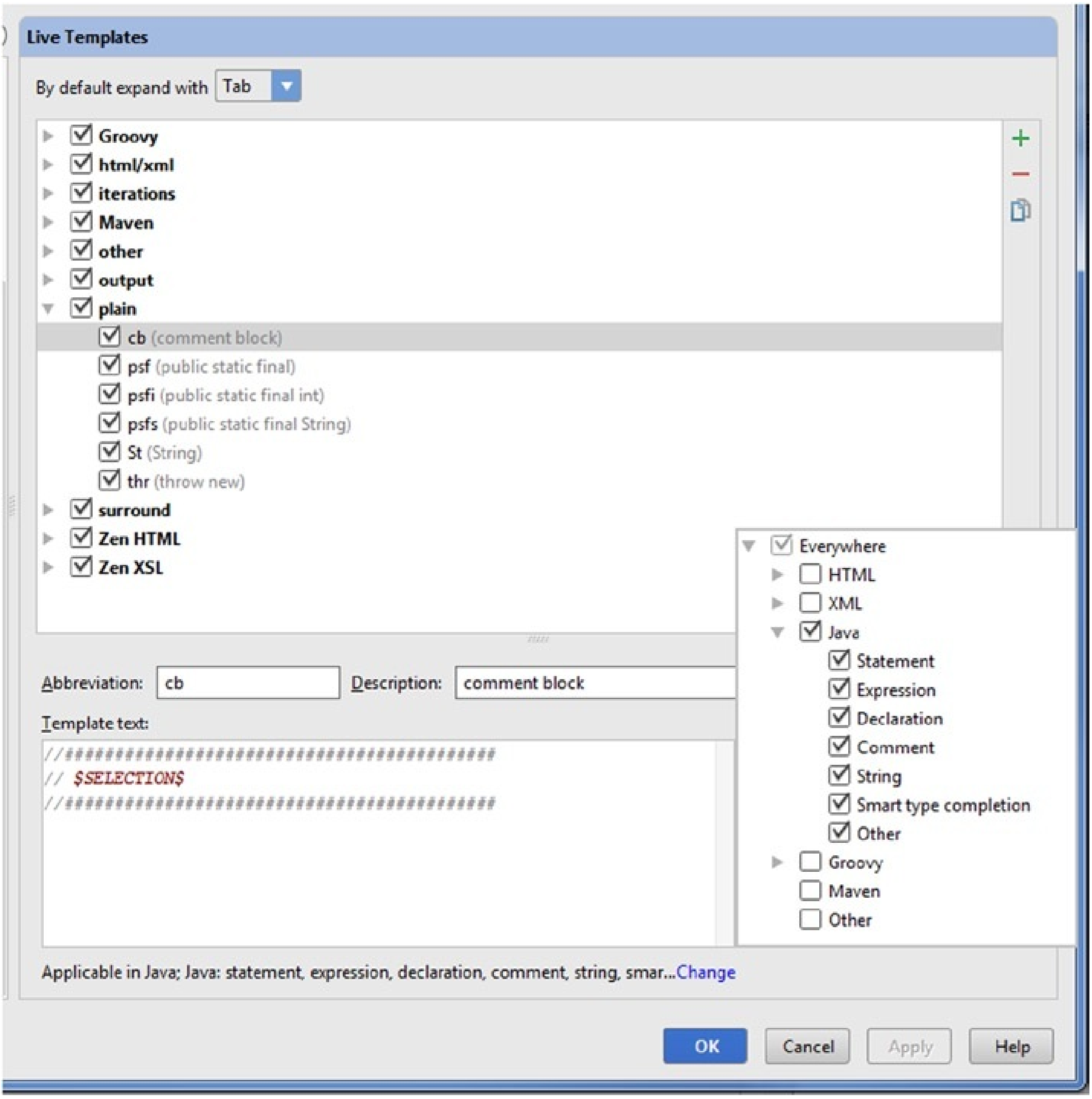
您刚刚创建一个称为 `cb` 的自定义动态模板,将在任何Java源文件和任何范围中编码时都是可用的。如图3-18，红色的字 `$SELECTION$` 是一个变量。很快你会看到这个变量在起作用。动态模板的选项如表3 - 4所示

Table 3-4. Live Template Options

Option	PC Keys	Mac Keys	Description
Insert Live Template	Ctrl+J	Cmd+J	Invoke a scope-sensitive Live Template list. Will insert template code into your document.
Surround with Live Template	Ctrl+Alt+J	Cmd+Alt+J	Invoke a scope-sensitive Surround with Live Template list. Will surround the selection with a scope-sensitive live template.

在离开动态模板的设置页面之前,浏览一下现有的动态模板，简称为 `psfs` ,在简单模板组中。单击psfs来查看其内容。你会注意到这个模板最终生成一个公共静态的字符串常量,它仅在Java和Groovy声明范围可用。单击OK返回Editor。在 `Sandbox` 的声明部分,在 `mGreetings` 的定义下面,键入 `psfs` 然后按 `Ctrl + J | Cmd + J` 调用动态模板,然后按Enter键。给这个常数一个名称来完成这个声明并赋值，如 `public static final String HELLO = “Hello Sandbox”;` 。

注意：在Java中,常量的命名约定是全部大写。在构造函数,输入 `CONSTRUCTORS` 这个单词。现在这个词转换成一个注释块好引起其他程序员的注意。选择整个词, `CONSTRUCTOR` ,然后按 `Ctrl + Alt + J | Cmd + Alt + J` 弹出动态住模板。从动态模板列表中选择 `cb` 和按 `Enter` 键,如图3-19。你即可应用在前面创建的动态模板



应用动态模板称为 `cb` (注释块)

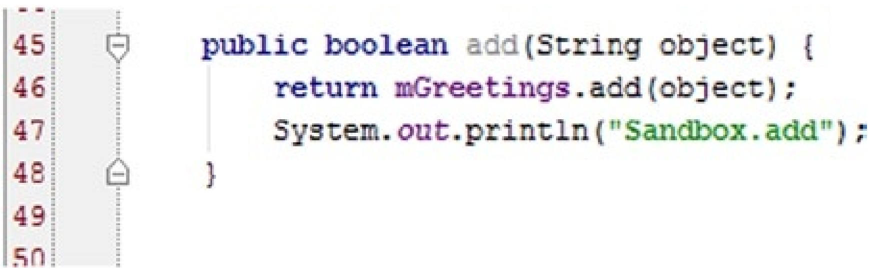
Moving Your Code

Android Studio 可以感知代码块结束,所以按行或者代码块移动都是很容易和直观的。移动语句和移动行之间的差别是：移动语句代表着考虑界限和范围,而行不考虑。如果你选择有着Move Statement的代码语句来移动，该语句将保留它的块的边界范围。如果你移动相同的有着Move Line的语句，Android Studio将语句作为一个简单的行文本,并将它移动到任何你想要到的地方。

你也可以移动整个代码块。 有了Move Statement,所有你需要做的是把光标放在你想移动的块的开始行(花括号的),按 `Ctrl + Shift + Down | Cmd + Shift +Down` 或者 `Ctrl + Shift + Up| Cmd + Shift + Up` ，在考虑其他的边界块和保持封闭的边界范围内的情

况下，整个块将集体移动。Move Line 不能识别范围或界限,但你仍然可以移动多行，通过 `Alt + Shift + Up | Alt + Shift +Down` （分别在PC和Mac中）实现Move Line Up or Move Line Down 操作。

要理解 Android Studio的移动操作,最好就是实践一下。首先让我们在add()方法中创建一个语句，随后直接返回 `mGreetings.add(object)` ,按回车键并开始一个新行并键入 `soutm` 。然后按 `Ctrl + J | Cmd + J` 调用动态模板,会产生 `System.out.println("Sandbox. add");` 。您可能已经注意到,你的新行代码将不会达到,因为返回语句在其上面,如图3-20所示。用Move Statement Up移动这个语句，同时按住 `Ctrl | Cmd` 和 `Shift` ,多次按向上箭头键。Android Studio 重新放置这个语句，但是不让你把这个语句移到一个可能没有任何意义的范围。再次尝试此操作 Move Line (`Alt + Shift +Up`),观察其行为。



Move Statement 和 Move Line

让我们尝试一下另一个例子来演示移动Move Statement的强大：把构造函数移动到类的底部。确保 `Sandbox()` 声明和上面的注释块之间没有空换行。现在,把你的光标在声明的 `Sandbox()` ，并且按住 `Ctrl | Cmd` 和 `Shift` 键，同时按下箭头键来调用Move Statement Down，重复直到你的构造函数是类的最后一个方法。注意,整个块,包括注释，都移动到类的底部了,避免了其他方法。移动代码操作和键盘快捷键表如3 - 5所示。

Table 3-5. Move Code Options

Option	PC Keys	Mac Keys	Description
Move Statement Down	Ctrl+Shift+Down	Cmd+Shift+Down	Moves a statement or statements down, within the boundaries of scope. If a block is moved, the entire block will leapfrog en masse to the next syntactically correct location.
Move Statement Up	Ctrl+Shift+Up	Cmd+Shift+Up	Same as Move Statement Down, but moves up.
Move Line Down	Alt+Shift+Down	Alt+Shift+Down	Moves statement(s) or line(s) down. Does not respect scope boundaries or syntax.
Move Line Up	Alt+Shift+Up	Alt+Shift+Up	Same as Move Line Down, but moves up.

样式代码

代码风格规范在不断发展。没有固定的规则，你应该在你的方法之后放置空格的数量，还是左括号应该出现在同一行作为方法签名或略低于它。组织倾向于定义自己的代码风格,但每个程序员的代码风格也各不相同,你也可能有你习惯的代码风格。幸运的是,Android Studio很简单就能样式化和组织你的代码。在开始样式化代码之前，让我们检查一下代码风格的设置。选择 `File ▸ Settings ▸ Code Style` 弹出设置对话框,如图3 - 21所示。Java和XML是我们在Android中最感兴趣的语言。在左窗格中切换打开代码风格选项,选择Java,并检查在设置窗口的每个选项卡。

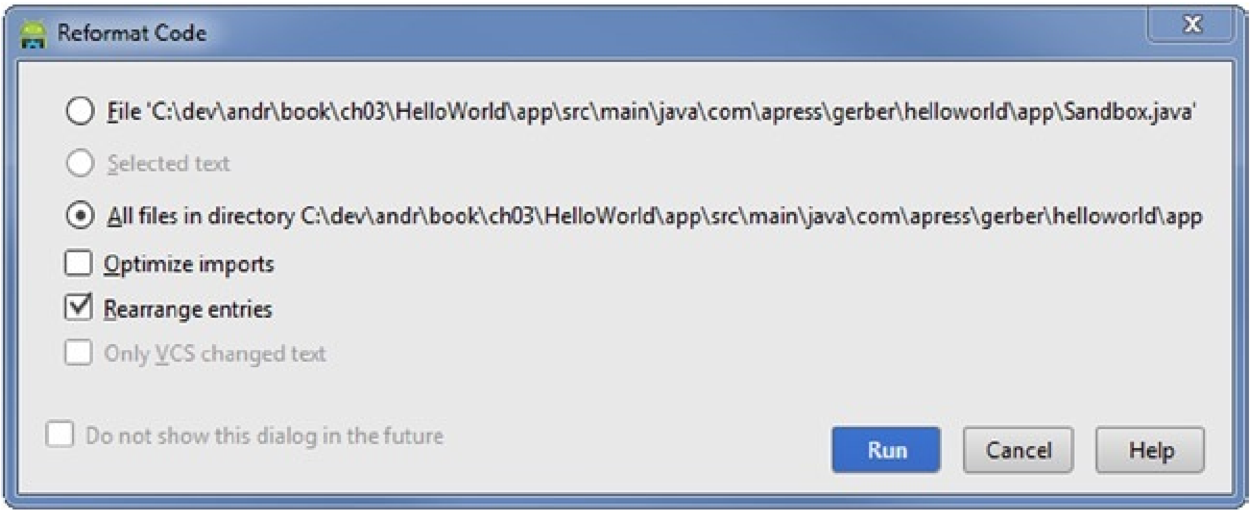


图3-21 代码风格➤Java selected并显示包装和括号选项卡的设置对话框

通过在中间的面板的各种tab页的这些设置选择/取消选择复选框的实验，注意类在右窗格中发生相应变化,满足你的风格。点击Manage按钮在顶部定义一个新的架构。现在点击另存为,给您的架构起一个名字,比如android,并单击OK。如果你做出进一步更改保存方案,单击Apply按钮应用它们。当你用 `Ctrl + Alt + L | Cmd + Alt + L` 格式化你的代码,在代码中设置你选择的风格标签将被应用。代码组织的选项如表3 - 6所示。

Table 3-6. Code-Organizing Options

Option	PC Keys	Mac Keys	Description
Auto-Indent Lines	Ctrl+Alt+I	Ctrl+Alt+I	Applies indents to the currently selected line or lines according to scheme settings.
Optimize Imports	Ctrl+Alt+O	Ctrl+Alt+O	Removes any un-used imports from import statements. Android Studio is so vigilant about keeping imports clean and relevant, that this command is practically redundant.
Rearrange Code	None		Re-arranges the order of code elements according to the rules established in the Arrangement settings.
Reformat Code	Ctrl+Alt+L	Cmd+Alt+L	Applies the code style settings for a particular scheme.

自动缩进行

自动缩进行有助于保持正常的行缩进代码。在Java中管理标签和缩进规则，通过 `File ➤ Settings ➤ Code Style ➤ Java ➤ Tabs and Indents` 。自动缩进线应用于当前行,或如果您有多个行选中,则是选中的所有行。

`Sandbox.java` ，选择整个方法的代码块,按 `Tab` 键。块应该向右移动一个选项卡的距离。现在把光标放在第一行的块,按 `Ctrl + Alt +I` （在PC和Mac中）。你会注意到自动缩进复位这条线到适当的缩进位置,尽管方法块的其余部分仍不受影响。现在选择的所有代码的类按 `Ctrl + I| Cmd +A` ，并且按 `Ctrl + Alt +I` 。这次,适当的缩进将会应用于整个文件。

重新整理代码

Arrangement管理代码中元素的顺序。例如,大多数人喜欢保持在类的顶部声明类成员,其次是构造函数,其次是 `getter` 和 `setter` ,等等。通过 `File ➤ Settings ➤ Code Style ➤ Java ➤ Arrangement` ，您可以编辑Arrangement的安排设置选项卡。在前面的部分中,您将构造函数的类。这不是通常他们在的地方。从主菜单 `Code ➤ Rearrange` 。你会注意到构造函数已经搬回其预期的位置——声明部分的下面。Rearrange Code按照 Arrangement 的设置执行这种重排操作。

代码重新格式化

重新格式化代码是在代码样式化中最强大的功能，因为它给了你选择应用的所有代码风格选项中定义的代码风格设置。正如您已经看到的,代码风格设置可以通过从主菜单中访问—— `File ➤ Settings ➤ Code Style` 。此外,重新格式化代码允许您重新格式化当前选择的文件,或相同类型的每个文件和目录。此外,重新格式化代码允许您链接重新排列条目(这将重新排列代码的Java文件),并优化导入到命令,如图3-22。按 `Ctrl + Alt + L | cmd + Alt + L` ，尝试重新格式化 `Sandbox.java` 。

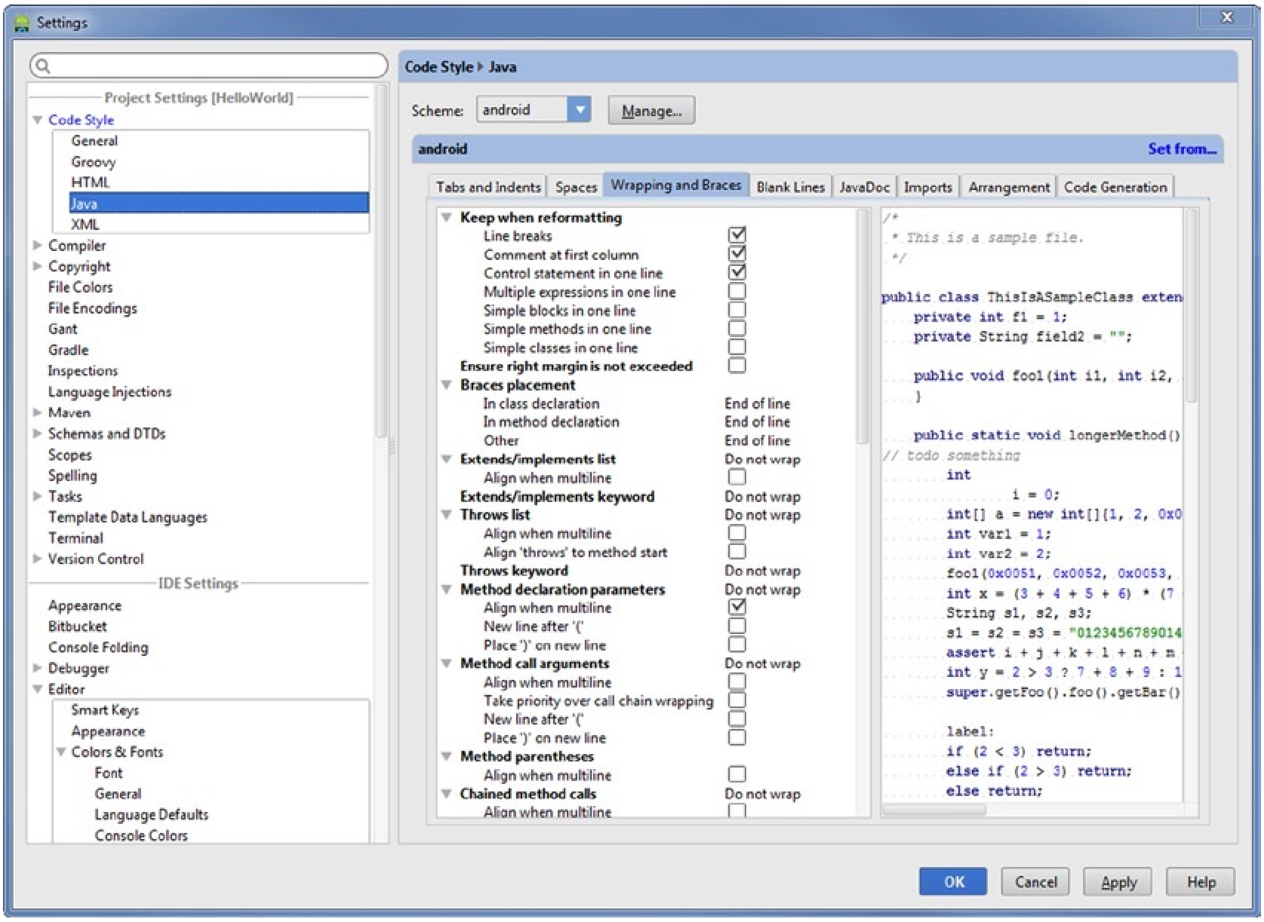


图3-2 包含重新整理条目选择的重新格式化代码对话框

环绕

环绕(`Ctrl + Alt + T | Cmd + Alt + T`)是一个在周围发现功能的超集，环绕动态模板(`Ctrl + Alt + J | Cmd + Alt + J`)。然而,环绕还包括选项来选择与Java语句或语句块周围如 `if / else` , `try / catch` 等等。虽然在你的 `Sandbox` 类中，简单的代码不可能产生任何受控异常,周围的语句可能抛出异常的 `try / catch` 块的周围是最好的应用,这可能是为什么键盘快捷键 `Ctrl + Alt + T | Cmd + Alt + T` 包括一个 `T` 。环绕的操作如表3 - 7所示。

Table 3-7. Surround With Options

Option	PC Keys	Mac Keys	Description
Surround With	Ctrl+Alt+T	Cmd+Alt+T	Surrounds the selected statement or statements with a Java code block such as if/else, for, try/catch, etc.
Unwrap/Remove	Ctrl+Shift+Delete	Cmd+Shift+Delete	Unwraps code blocks from selected statement or statements.

在 `Sandbox.java` 的 `add()` 方法中，您希望确保没有重复的。让我们用一个 `if / else` 块围绕 `return mGreetings.add(Object);` 如图3-23。选择整行,按 `Ctrl + Alt + T | Cmd + Alt + T` 激活周围。现在从菜单中选择 `if / else` 。在 `if` 的括号里键

入 `!mGreetings. contains(object)` ，在 `else` 块返回 `false;` 。

```
public boolean add(String object) {
    System.out.println("Sandbox.add");
    if (!mGreetings.contains(object)) {
        return mGreetings.add(object);
    } else {
        return false;
    }
}
```

图3-23 用Surround With包装和打开代码块

假如你的业务规则已经改变,你不再关心 `mGreetings` 重复的条目了。使用Unwrap/Remove删除刚才创建的 `if / else` 块。把你的光标在 `return mGreetings.add(object);` 语句的任何位置,按 `Ctrl + Shift +Delete| Cmd + Shift +Delete` ,并选择打开。该方法现在看起来应该像之前你修改它的那样。 环绕的另一个伟大的应用是遍历集合。在前面的小节中自动生成 `toString()` 方法。现在改变这种方法遍历 `mGreetings` 集合。删除 `toString()` 方法的返回语句，确保 `toString()` 方法的主体是空的。现在键入 `mGreetings` ,然后按 `Ctrl + Alt + T | Cmd + Alt + T` 。从列表中选择迭代Iterable,或者按 `I` 键。再次按下 `Enter` 接受 `greeting` 作为单个元素的名称。生成的代码是一个 `for - each` 循环。注意到Android Studio明白 `mGreetings` 包含字符串,而且它也产生一个局部变量叫做 `greeting` —— `mGreetings` 缺少 `m` 的单数形式。进一步修改 `add()` 方法,如图3-24。

```
@Override
public String toString() {

    StringBuilder stringBuilder = new StringBuilder();
    for (String greeting : mGreetings) {
        stringBuilder.append(greeting + " ");
    }
    return stringBuilder.toString().trim();
}
```

小结

本章涵盖了Android Studio最重要的代码生成功能。我们鼓励您返回 `File ➤ Settings ➤ Code Style ➤ Java and File ➤ Settings ➤ Code Style ➤` ，并花几分钟探索各种设置。Android Studio为编码提供了很多快捷键,但你不必记得。如果你会不知所措,你可以使用本书作为参考,或者导航到Code菜单和探索其菜单项和子菜单作为参考。

勘误感谢：Sohnyi ，zhin

第四章：重构代码

> 译者：qiangxcn
>
> 来源：Learn Android Studio 汉化教程 第四章 : Refactoring Code

在Android Studio中开发，解决方案不会总是一蹴而成的。作为一个有效率的编程者，在你的开发，调试和测试中需要一些弹性以及代码重构。随着在这章中的行进，你将明白Android Studio如何产生代码；在这章里你将看到Android Studio如何重构你的代码。重构代码最大的风险是可能引入不期望的错误。通过分析某些风险重构操作的结果，Android Studio减低了这些风险，然后激活Find tool窗口，在发行前，你可以预览你的更改-那里标志着任何错误或冲突。

在这章里描述的许多重构操作也可不用Android Studio的重构工具来执行。无论怎样，你必须避免强行重构（例如：通过一个全文的查找/替换操作来重组代码），因为在这些情形下Android Studio不能从引入的错误中一直保护你。相反，如果Android Studio发现你企图一个重构操作，它将尽力避免你犯任何愚蠢错误。例如，在Project tool窗口中拖动一个JAVA源文件从一个包到另一个包，这将强制产 `Refactor ➤ Move` 操作，由此会分析你的移动操作结果，允许你预览那些改变，然后适当地改变类中所有的 `import` 语句以适应合格的新全包名。

绝大多数的重构操作被限制于一个方法或一个类里面，因而将不太会引入错误到你的项目中。危险的重构操作是那些涉及到两个或以上的资源文件。如果重构操作引入编译错误，检查管理器将在编辑器里用红标签标志那些被影响的资源文件。基于这点，你或者可以企图去修正它们，或者用 `Ctrl+Z|Cmd+Z` 简单地取消这个重构操作了。如果重构操作以没有编译错误完成，然而涉及到太多资源文件,那你还是必须要测试及校验你是否引入了任何实时错误。第11章涵盖了测试。

提示：你必须提交任何重要的重构更改，作为单一的Git提交，因此之后你可以轻松地回复那个提交。第7章涵盖了Git。

这章聚焦于用最好的工具来执行重构操作。在我们探访独立的重构操作之前，我们乐于指出Android Studio有一个极其便利的重构工具叫 `Refactor ➤Refactor This` .总在上下文菜单中选择这个，图4-1所示，集成了绝大多数有用的重构操作。快捷键是 `Ctrl+Alt+Shift+T | Ctrl+T` ,在PC上你可以方便地记首字母CAST。

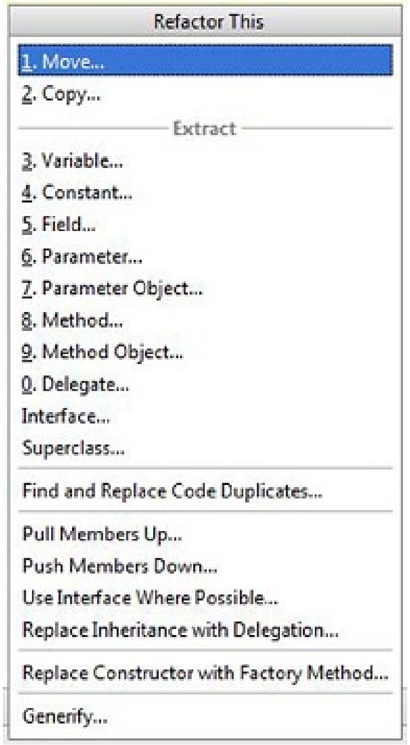


图4-1：重构器Refactor This菜单，包括了大部分的有用重构操作。

在你开始本章的例子之前，修改第3章的 `Sandbox.java` 文件，它没扩充和包含任何方法或成员变量，象下面的小片段：

```
public class Sandbox {
}
```

重命名

在Project tool窗口中选择Sandbox并导航到 `Refactor➤ Rename` 或按 `Shift+F6`.引出一个对话框以允许你重命名你的类，另外也可以重命名出现在注释中，测试实例，以及继承类中的那些名字。重命名 `Sandbox` 为 `Playpen` 并点击Refactor按钮，如图4-2所示。你将看到在你项目中重命名的操作。现在请用 `Ctrl+Z | Cmd+Z` 取消这个操作。

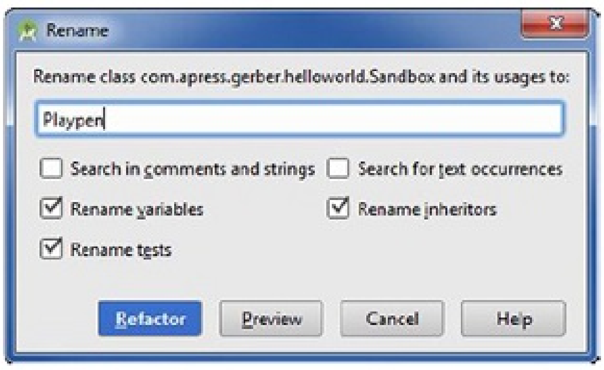


图4-2：重命名Sandox为Playpen

修改签名

更改签名的操作允许你改变方法的下列属性：可视性，名称，返回类型，参数，以及例外抛出。在 `Sandbox.java` 里创建一个方法，如下面的代码片断：

```
public String greetings(String message){
    return "Hello " + message;
}
```

将光标移到单词 `greetings` 上面（高亮的粗体字）并按 `Ctrl+F6 |Cmd+F6` ，或导航到 `Refactor ➤ Change Signature` 。结果对话框会允许你修改方法的签名，如图4-3所示。

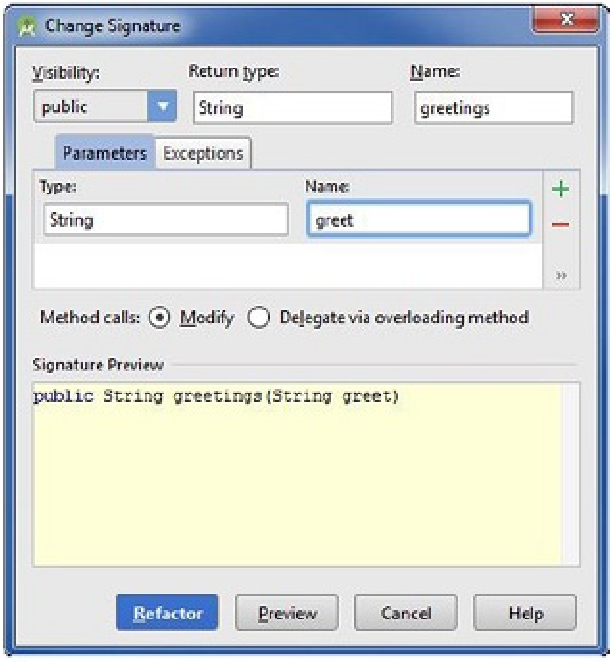


图4-3：修改签名对话框

在参数表里，点击字符串类型参数 `message` 这项，将它的名称从 `message` 改为 `greet` ，如图4-3所示。图标绿色的 `+` 和红色的 `-` 允许你分别增加或减少方法的参数。并且，你也可以编辑参数表里面它们的类型和名称。另外，修改当前方法时，你可能决定选择委托重载方法的单选按钮。选择这个按钮将保留你的原始方法不受影响，但会生成另一个你新定义签名的方法。在JAVA里如果你可能考虑一系列有相同的名称重载方法，但参数的顺序和/或参数的类型不同。无论如何，你做的更改并没限制方法的重载。如果你希望的话通过点击预览按钮，可以在提交它们之前预览。按Refactor按钮完成并离开。

类型移植

如标题所述，类型移植允许你从一个JAVA类型移植到另一个。让我们假设你创建了一个 `Person` 类。随着你的开发进行，你发现 `Person` 太一般了，因而你又创建了一个 `Manager` 类扩展于 `Person` 。如果你想移植所有的 `Person` 实例到 `Manager` ，你可以用类型

移植轻松办到。

将光标放到 `greetings` 方法的 `String` 声明上（在下面代码片段中的高亮粗体字）并且按 `Ctrl+Shift+F6 | Cmd+Shift+F6` 或选择 `Refactor ➤ Type Migration` 。结果对话框如图4-4所示。

```
public String greetings(String greet){
    return "Hello " + greet;
}
```



图4-4：类型移植，从 `String` 到 `date`

更改 `java.lang.String` 到 `java.util.Date` ，如图4-4所示。从选择范围框的下拉菜单里选择打开文件。如同大多数重构操作，通过点击预览按钮，你可预览你的更改。这里我们按Refactor按钮。

移动

你可以通以下三种方法中的一个方法来移动源代码文件。

- 在Project tool窗口拖动源文件从一个包到另一个包中；
- 选择源文件从主菜单导航到 `Refactor ➤ Move` ；
- 在Project tool窗口选择源文件并按下 `F6` 。

右键点击(在Mac电脑上 `Ctrl` -click) `com.apress.gerber.helloworld` 包并选择 `New ➤ Package` 。给这个重构目标包命名。从Project tool窗口，把 `sandbox.java` 拖放到重构目标包里，当出现图4-5对话框时按OK。在Project tool窗口中执行的任何拖放操作将自动产生一个重构移动操作，这个会让你安全地把一个类从一个包移动到别一个包里。

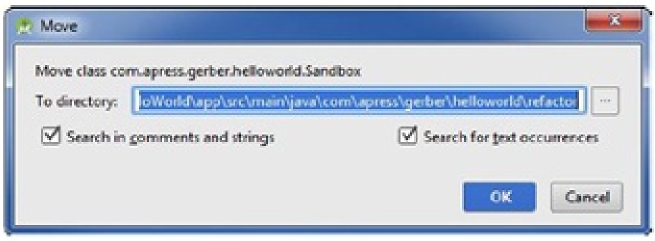


图4-5：由于拖放操作导致的重构移动对话框

除了移动类之外，你也可以移动成员变量。在你的 `Sandbox.class` 中，定义一个新成员变量如下：

```
public static final String HELLO = "Hello Android Studio";
```

移动光标到这行代码上并按下F6。导致一个对话框允许你移动成员变量从一个类到另一个。如图4-6所示。请点击取消按钮来取消这次操作。

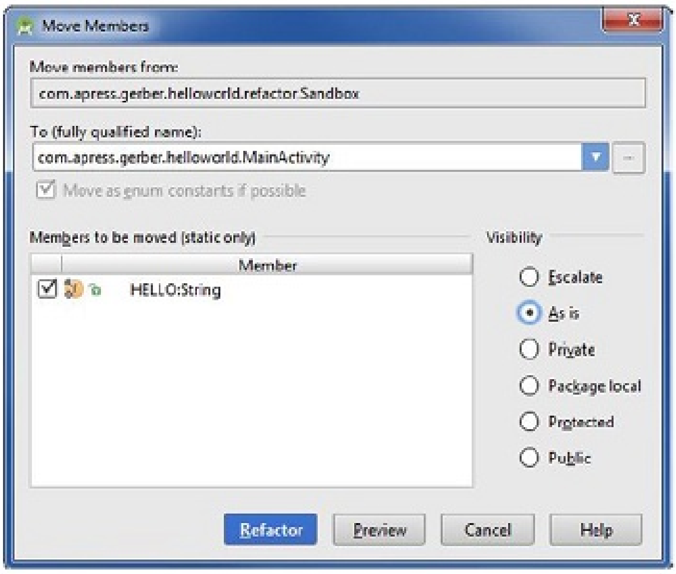


图4-6：移动成员变量的对话框

复制

复制，有点象移动，按快捷键 `F5` 或选择主菜单里的 `Refactor ‣ Copy` 来访问。在Project tool窗口，选择之前重构的包中的 `Sandbox.java` 并按下 `F5` 键。在目标包的拖放菜单中选择 `com.apress.gerber.helloworld` 包并点击OK，如图4-7所示。象我们这里无差别地复制JAVA源文件不是一个好主意，因为结果不明确且普通存在着错误。

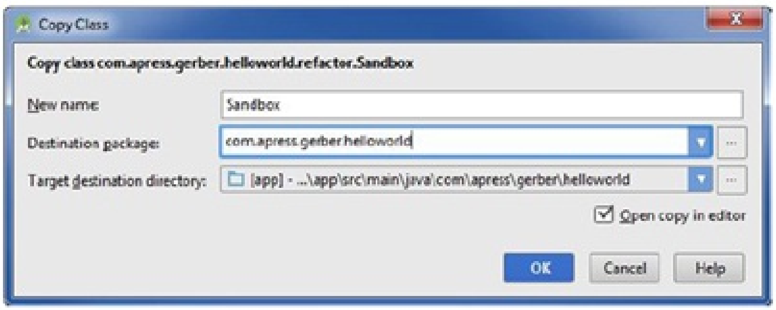


图4-7：复制类的对话框

安全删除

让我们来删除刚才复制的类。在Android Studio的Project tool窗口里用Delete按键,你总可以删除文件和资源。在刚才重构包中点击 `Sandbox.java` 文件并按下Delete键。结果对话框允许你使用安全删检查选项来删除。安全删除的先进之处在于执行删除前我们可查询任何因依赖于这个资产而可能导致的破坏，如图4-8所示。如果在这个项目中发任何东西依赖于这个资产，将给出一个可选项浏览它们，或点击无论如何都删除可选项来强制删除。



图4-8：安全删除对话框

析出

析出不是一个操作而是几个。这节涵括了一些重要的析出操作：析出变量，析出常量，析出域，析出参数，和析出方法。在·Sandbox.class·里，让我们先移走所有的成员变量和方法从白纸开始：

```
public class Sandbox {

}
```

析出变量

在你的 `Sandbox.java` 类里，定义一个方法如下示：

```
private String saySomething(){
    return "Something";
}
```

把光标放到硬编码Something的值（粗体字）上并选择 `Refactor ‣ Extract ‣ Variable` ，或者按 `Ctrl+Alt+V | Cmd+Alt+V` 接着按回车，不要选择声明 `final` 的复选框。Android Studio依据硬编码字符串析出一个本地变量并命名它。你将以如下代码告终：

```
private String saySomething(){
    String something = "Something";
    return something;
}
```

析出常量

当你开发Android的APP时，会发现你会用很多的字符串作为键-例如，在 `Map` 和 `Bundle` 里。因此，析出常量将会节省你大量的时间。

定义一个方法象如下的代码片段。把光标放到 `name_key` 字符串上并按 `Ctrl+Alt+C | Cmd+Alt+C` 。结果对话框将如图4-9所示。这里，Android Studio提供一些建议的名称。按惯例，常量在JAVA里都应该是大写的。选择 `NAME_KEY` 并按回车。 注意：为了创建和处理这个方法而不发生编译错误，你必须导入 `android.os.Bundle`

```
private void addName(String name, Bundle bundle ){
    bundle.putString("name_key", name);
}
```

结束时你将看一个常量名称是 `NAME_KEY` ，象以下定义的：

```
public static final String NAME_KEY = "name_key";
```

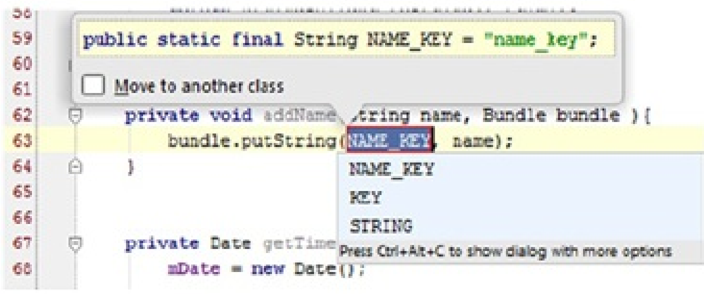


图4-9：析出常量NAME_KEY

析出域

析出域将转换一个本地变量到你的类域（a.k.a.成员变量）里。 注意：为了创建和处理这个方法而不发生编译错误，你必须导入 `android.os.Bundle` 在你的 `Sandbox` 类里定义一个方法：

```
private Date getDate(){
    return new Date();
}
```

把光标放到 `Date` （粗体高亮）上并按键 `Ctrl+Alt+F | Cmd+Alt+F` 。你将看到一个对话框如图4-10所示。在Android里，按惯例域(a.k.a. 成员变量)名的头字母用 `m` 。你将会注意到一个下拉菜单允许你初始化当前方法中的域，域声明，或构造器。选择域声明并按回车。

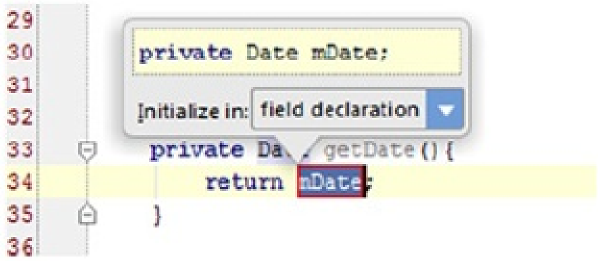


图4-10：析出域对话框

结果如下：

```
private final Date mDate = new Date();
...
private Date getDate(){
    return mDate;
}
```

移除 `final` 关键字，结果这个域声明如下：

```
private Date mDate = new Date();
```

析出参数

析出参数允许你析出一个变量并放到封套的方法里。在你的 `Sandbox` 类里定义一个方法：

```
private void setDate(){
    mDate = new Date();
}
```

将光标放到Date()（粗体高亮）上，`按Ctrl+Alt+P | Cmd+Alt+P` 接着按回车。结果这个方法如下代码片段所示：

```
private void setDate(Date date){
    mDate = date;
}
```

析出方法

析出方法让你选择一行或多行连续的代码放到一个分离的方法中。有两个原因你会想做这个。首先是方法太复杂了。相对于100行的代码，打断它到约10到20行的算法块会比较容易阅读且会较少错误。

再者，重复的代码从来就不是好主意，因而如果你发现了重复的代码，最好析出它们成一个方法并在重复的地方调用它。通过析出之前重复的代码到一个方法里并调用它，你可以保留你的方法在一个地方，如果你想修改它，你只要修改一次。在你的 `Sandbox` 类里重建下面的两个方法如清单4-1。你可以自由地复制和粘贴。

清单4-1

```
private String methodHello (){
    String greet = "Hello";
    StringBuilder stringBuilder = new StringBuilder();
    for(int nC = 0; nC < 10; nC++){
        stringBuilder.append(greet + nC);
    }
    return stringBuilder.toString();
}
private String methodGoodbye (){
    String greet = "Goodbye";
    StringBuilder stringBuilder = new StringBuilder();
    for(int nC = 0; nC < 10; nC++){
        stringBuilder.append(greet + nC);
    }
    return stringBuilder.toString();
}
```

象我们已经提到过的，任何时候你发现重复一个代码块或复制和粘贴代码块，你就必须考虑析出方法了。选择清单4-1中粗体高亮的块。现在按 `Ctrl+Alt+M | Cmd+Alt+M` 析出方法。将呈现一个对话框列出方法的签名。重命名这个方法为 `getGreet` ，如图4-11所示，接着点击OK。

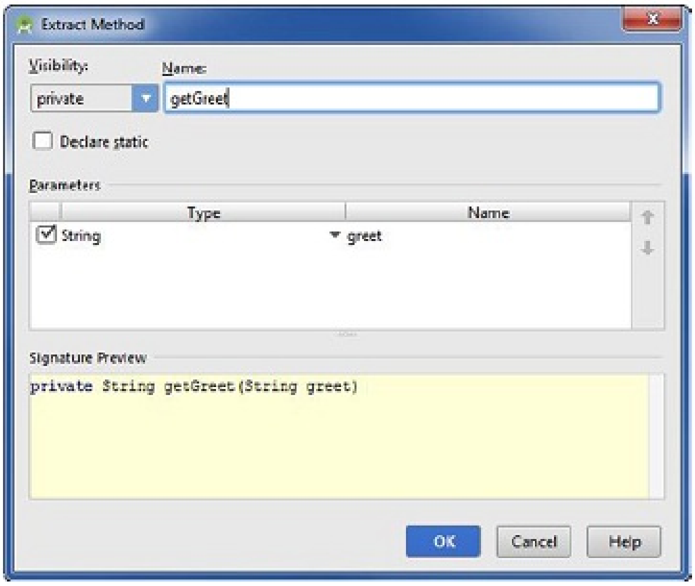


图4-11：析出方法对话框

Android Studio扫描你的文件并看到你有另外一个确凿的代码块案例。在处理复制对话框里点击Yes接受建议，如图4-12所示。

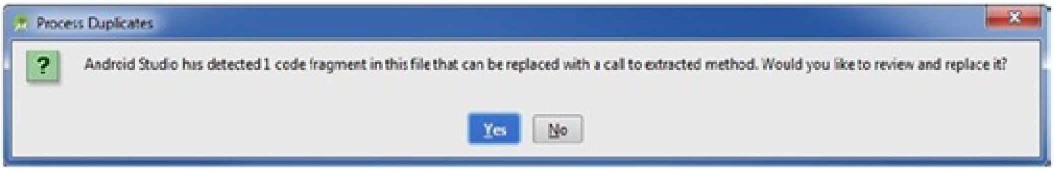


图4-12：复制处理对话框

你将看到清单4-2的代码。现在结果方法很容易保留在某个地方。

清单 4-2.从析出方法操作里得出的结果代码

```
private String methodHello (){
    String greet = "Hello";
    return getGreet(greet);
}
private String methodGoodbye (){
    String greet = "Goodbye";
    return getGreet(greet);
}
private String getGreet (String greet){
    StringBuilder stringBuilder = new StringBuilder();
    for(int nC = 0; nC < 10; nC++){
        stringBuilder.append(greet + nC);
    }
    return stringBuilder.toString();
}
```

高级重构

整个这章所描述的剩下的其他重构操作是高级别的。如果你有兴趣简单迅速地提高Android Studio，已经有了足够的知识来有效地使用重构操作，也许你可以跳过这节。但无论如何，如果你对JAVA有比较好的理解并且想钻研更多的重构操作，继续看下去吧。

移除Sandbox.java里的所有成员变量及方法，从一张白纸开始。

```
public class Sandbox {

}
```

右键点击（Mac电脑用 `Ctrl` +点击）Project tool窗口里的 `com.apress.gerber.helloworld` 包，选择 `New ➤ Java Class` 。命名这个类为 `Minibox` 。修改Minibox类定义继承自 `Sandbox` 并且有个成员变量叫做 `mShovel` ，如这里所示：

```
public class Minibox extends Sandbox {
    private String mShovel;
}
```

推高成员变量以及拉低成员变量

推高成员变量以及拉低成员变量用于继承。注意到我们定义了一个 `mShovel` 成员变量在 `Minibox` 类里。让我们假设 `mShovel` 可能对于其他继承自 `Sandbox` 类的子类也有用。这样做，打开 `Minibox` 类并选择 `Refactor ▸ Pull Members Up` 。结果对话框如图4-13所示。

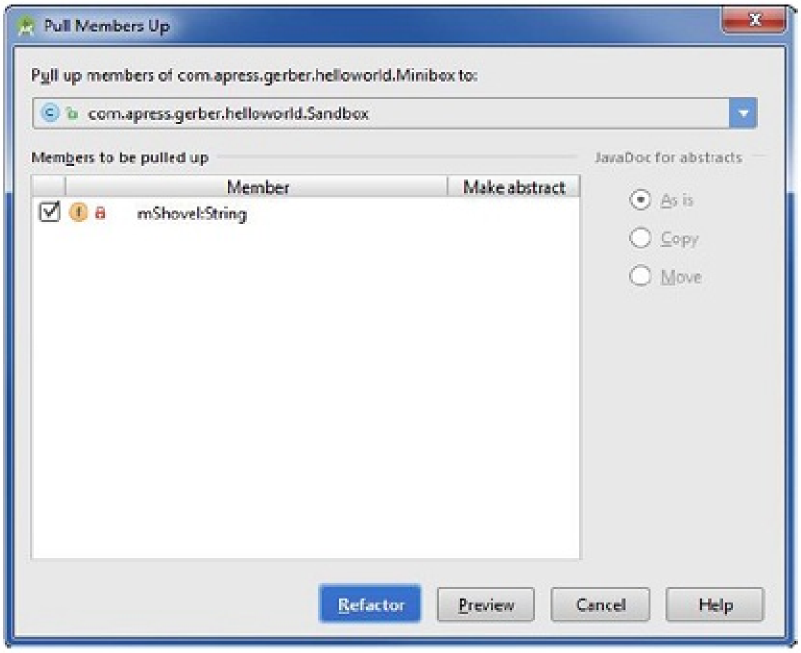


图 4-13： 推高成员变量对话框

`mShove` l会被缺省地选中，并且因为 `Sandbox` 是 `Minibox` 的父类所以推高成员变量组合窗口会缺省地设定在 `com.apress.gerber.helloworld.Sandbox` 类上。点击Refactor，检视 `Sandbox` 和 `Minibox` ，你将发现 `mShovel` 成员变量不再出现在 `Minibox` 里了，而属于 `Sandbox` 。作为一般规则，如果你觉得一个成员变量对其他扩展类也是有用的，那么就把这些成员变量在继承层次里推高。用类似的手法可以拉低成员变量。

解除继承并委托成员变量

右键点击（Mac电脑用 `Ctrl` +点击） `com.apress.gerber.helloword` 包选择 `New ▸ Java Class` 。命名这个新类为 `Patio` 并继承自 `Sandbox` ：

```
public class Patio extends Sandbox {

}
```

进一步分析后，我们决定 `Patio` 不是一个 `Sandbox` ，但更象是它拥有一个 `Sandbox` 。为了改变它们的关系，导航到 `Refactor ▸ Replace Inheritance with Delegation` 。在结果对话框里，点击为委托成产生 `getter` 方法，如图4-14所示。

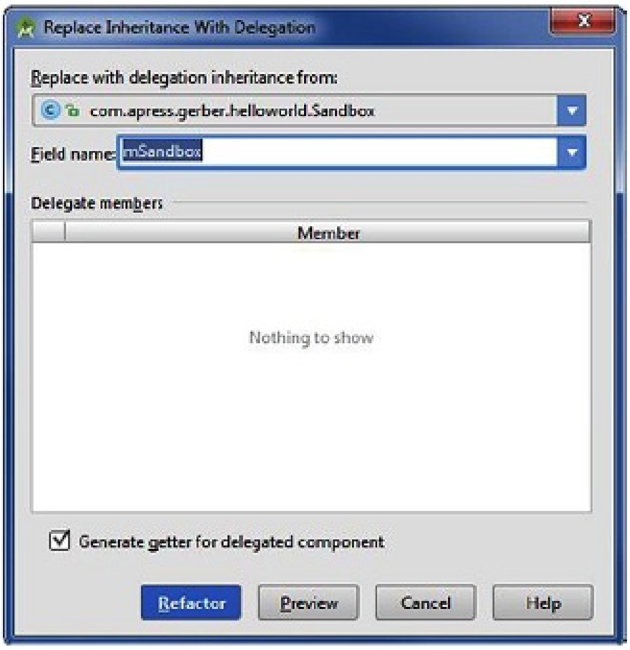


图 4-14：解除继承并委托成员变量对话框

现在你的 `Patio` 类将拥有一个 `Sandbox` 成员变量，如下代码片段所示：

```
public class Patio {
    private final Sandbox mSandbox = new Sandbox();
    public Sandbox getSandbox() {
        return mSandbox;
    }
}
```

封装域

封装域是一种面向对象策略，通过使访问级别为私有来隐藏类成员变量并提供以公开的 `getter/setter` 方法作为公开的接口。尽管当你选择 `Refactor ➤ Encapsulate Fields` 时有很多的选择，但 `Refactor ➤ Encapsulate Fields` 类似于 `Code ➤ Generate ➤ Getter and Setter` 。打开你的 `Sandbox` 类并定义一个成员变量叫 `mChildren` ，如下面代码片段中的粗体高亮部分。从主菜单，选择 `Refactor ➤ Encapsulate Fields` 。

```
public class Sandbox {
    private String mShovel;
    private int mChildren;
}
```

结果对话框允许你精确地选择怎样封装这些域以及它们的访问级别。真正地封装域将有一个私有的可视度及公有的访问（ `getter` ）及改变（ `setter` ）方法。点击Refactor按钮，如图4-15所示，注意到Android Studio在我们的 `Sandbox.java` 类里为我们生成所有的 `getter` 和 `setter` 。

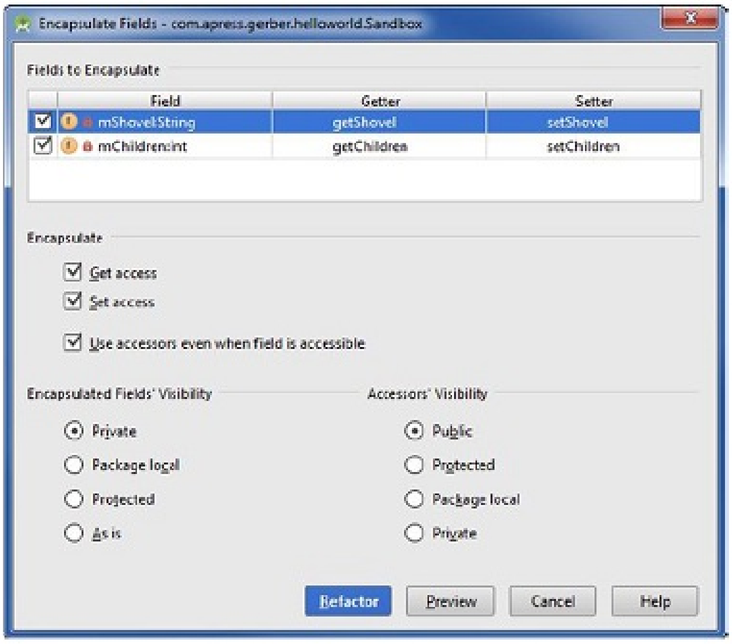


图 4-15：封装域对话框

包装方法的返回值类型

当你需要返回一个对象而不是基本数据类型时，包装返回值可能是有用的（当然还有其他的情形你可能想包装一个返回值）。把光标放到 `getChildren()` 方法上并导航到 `Refactor ➤ Wrap Method Return Value` 。选择Use Existing Class选项，名称框输入 `java.lang.Integer` ，包装域框则选择value，如图4-16所示。现在点击Refactor按钮且注意到你的 `getChildren()` 方法返回一个 `Integer` 类，而不是基本数据类型 `int` 。

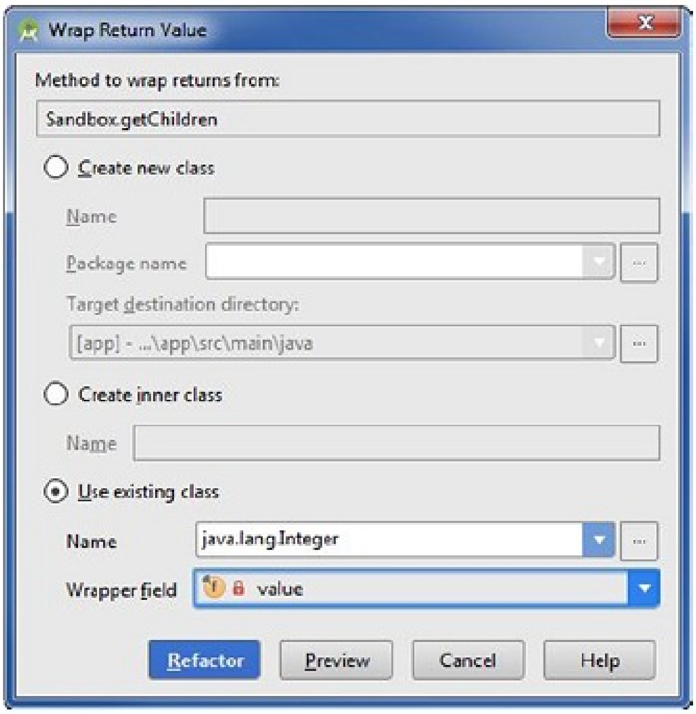


图 4-16： 包装返回值对话框

用工厂方法代替构造器

把光标放在 `Sandbox` 类定义附栏内。按 `Alt+Insert | Cmd+N` 接着选择生成一个新构造器。选择所有成员变量，如图4-17所示，然后点击OK。

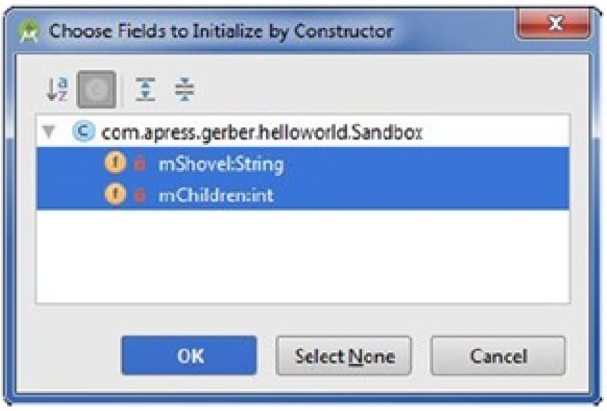


图 4-17：通过构造器来选择域的对话框

将光标放到如下所示代码片段构造器的任何位置，并导航到 `Refactor ➤ Replace Constructor with Factory Method` 。结果对话框象图4-18所示。点击Refactor来生成一个工厂方法。

```
public Sandbox(String shovel, int children) {
    mShovel = shovel;
    mChildren = children;
}
```



图 4-18： 用工厂模式代替构造器的对话框

注意到之前的构造器现在变成 `private` 了并且一个新的静态方法返回 `Sandbox` 类的实例，如下面代码片段所示。如果你正创建一个单例类，这将会是非常有用的。

```
public static Sandbox createSandbox(String shovel, int children) {
    return new Sandbox(shovel, children);
}
```

转换匿名内部类

在你的Sandbox类的构造器里加上下面行：

```
new Thread(new Runnable()).start();
```

把光标放在 `Runnable()` 上并按下 `Alt+Enter` 来调用完成代码操作。然后选择实现方法。选择 `run()` 方法并点击OK。你的代码将象下面片段所示：

```
new Thread(new Runnable() {
    @Override
    public void run() {
        //do something
    }
}).start();
```

把光标放在 `Runnable()` 上并导航到 `Refactor ‣ Convert Anonymous to Inner` 。Android Studio建议 `MyRunnable` 作为这个类名，如图4-19所示。去选Make class static选项后点击OK。注意到你将在 `Sandbox` 类里得到一个私有的内部类叫 `MyRunnable` ，它实现 `Runnable` 接口。这个例子没有做更多的事；不管怎样，当委派 `View` 的行为时你也许有机会用到这个。

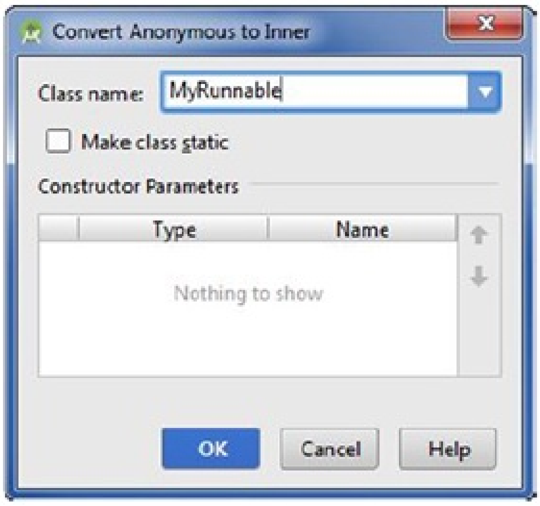


图 4-19： 转换匿名内部类的对话框

小结

这章讨论了Android Studio目前有效的许多重构操作。重构代码是编程工程的一个必需的部分，Android Studio里的重构工具是最好的重构工具之一。Android Studio通过分析推理以及允许你于提交操作前在Find tool窗口预览结果，减低了某些操作执行的风险。最重要的是重构操作 `Refactor ‣ Refactor This` 对话框是如此有效的，这个可以通过用快捷键 `Ctrl+Alt+Shift+T | Ctrl+T` 来调用。

第五章：Reminders实验：第一部分

> 译者：qiangxcn
>
> 来源：
>
> - Learn Android Studio 汉化教程 第五章：Reminders实验：第一部分
> - Learn Android Studio 汉化教程 第五章：Reminders实验：第一部分（续）

到现在为止你已经熟悉了创建一个新项目，编程，和重构的基本操作了。是时候创建一个Android 应用了，或者说成我们所谓的APP。这章将介绍四个实例项目的第一个。这些实例目的是让你熟悉使用Android Studio开发APP的上下文方面。核心功能是允许你创建和删除一个提醒以及标志那些重要的提醒。重要的条目文体左边将被强调黄色标签。这个APP将合同一个动作条菜单，上下文菜单，一个本地的数据库作存储，如果设备支持的话还有多重选择。

图5-1图示了仿真器上完成的app。这个例子介绍了Android基础并且你也会学到如何用内建的SQLite数据库来保存数据。别担心那些你不熟悉的话题，后续的章节将涵括这些话题大量的细节。

注意：为了一致性，我们建议你用Git克隆这个项目，虽然你将从草稿里用它自身的Git存储库重建这个项目。如果你还没有安装Git，请看第7章。在窗口里打开一个Git-bash会话（在Mac或Linux里的终端）并导航到 `C:\androidBook\reference\` （译者注：在Git-bash控制台里可以使用Dos命令 `cd c:` 改当前盘为 `c` ; `dir` 查看当前目录文件; `cd androidBook` 进入 `androidBook` 子目录； `cd reference` 进入 `reference` 子目录）（如果没有这个目录创建它）并提交下面的Git命令： `git clone https://bitbucket.org/csgerber/reminders.git Reminders` 。

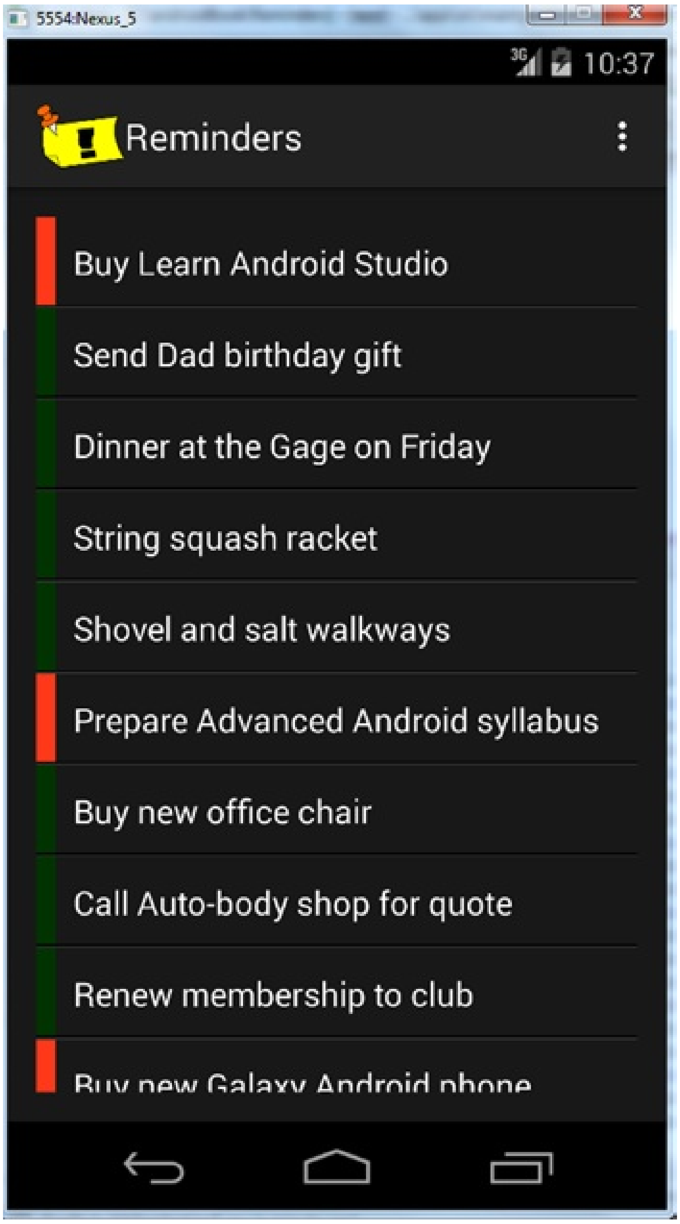


图5-1 完成的app界面

操作Reminders这个app，你可以用动作条的溢出菜单。点击溢出按钮，在菜单栏右侧，看起来象垂直点，打开一个菜单如图5-2有两个选项：新提醒，和退出。点击新提醒打开一个对话框如图5-3。在这个对话框里，你可以为新提醒加入文本并且点提交加到清单里。点击退出。

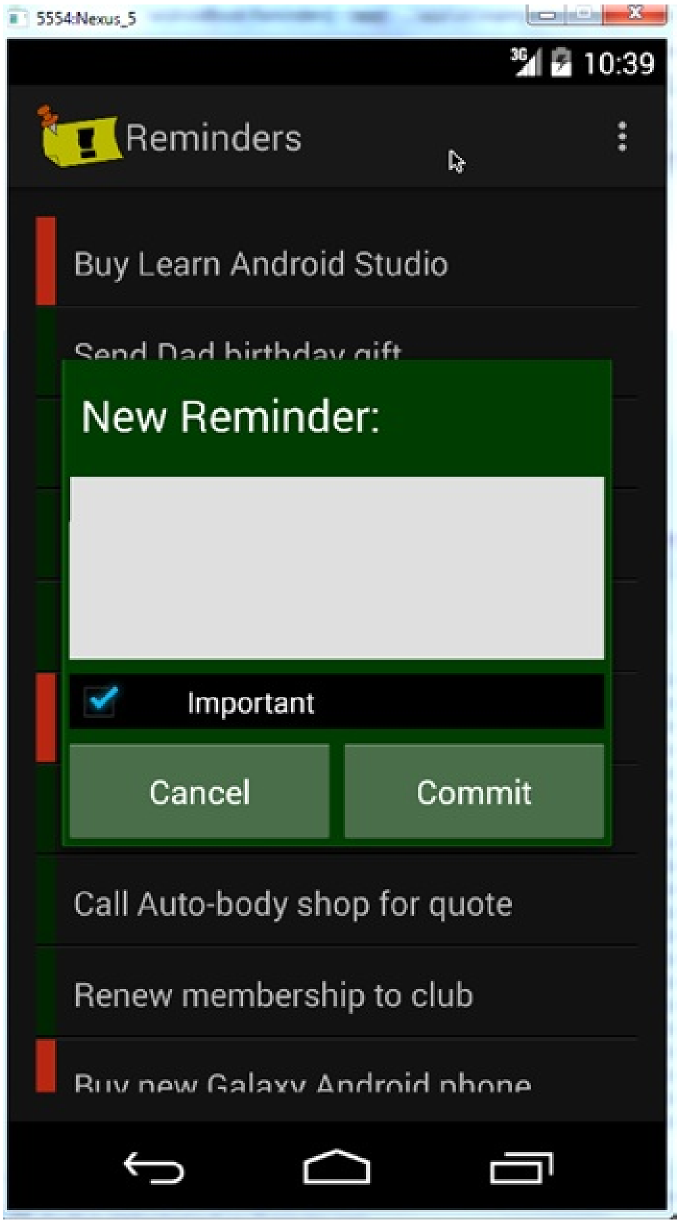


图5-2 激活溢出菜单的app界面

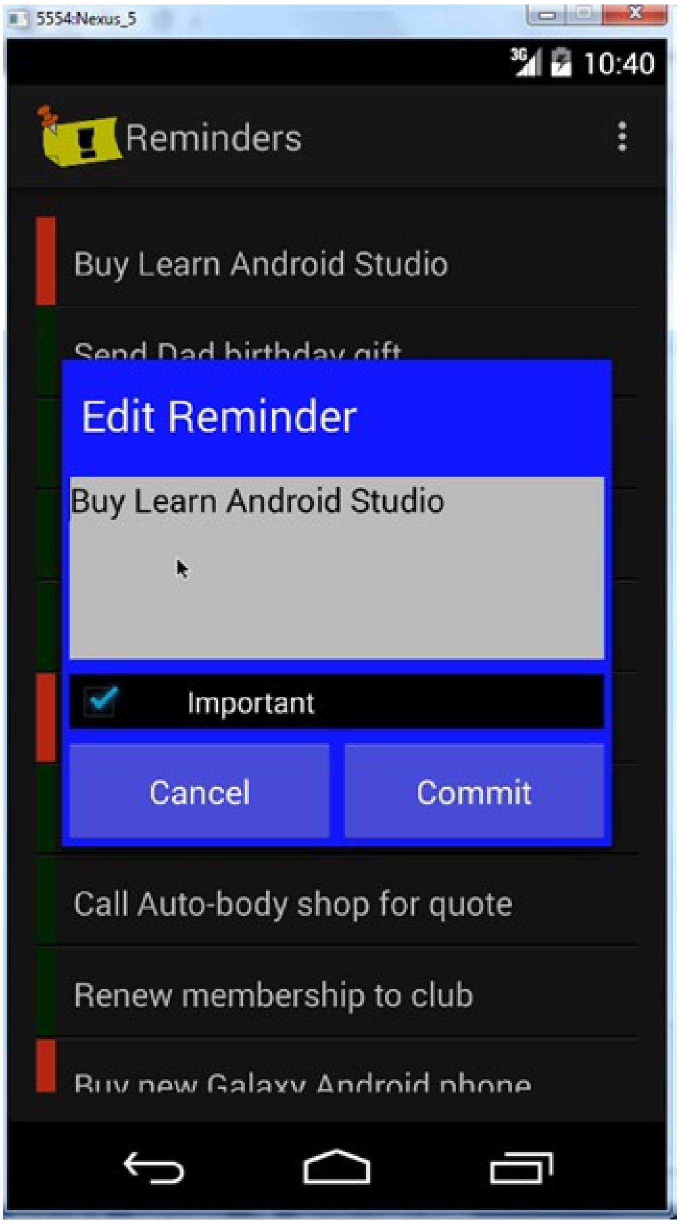


图5-3 新提醒对话框

点击清单中的随意一个提醒有两个可选方式，如图5-4：编辑提醒和删除提醒。从上下文菜单中点编辑提醒，弹出对话框如图5-5所示，在那里你可以修改提醒里的文本。点击删除提醒则会删掉这个提醒。

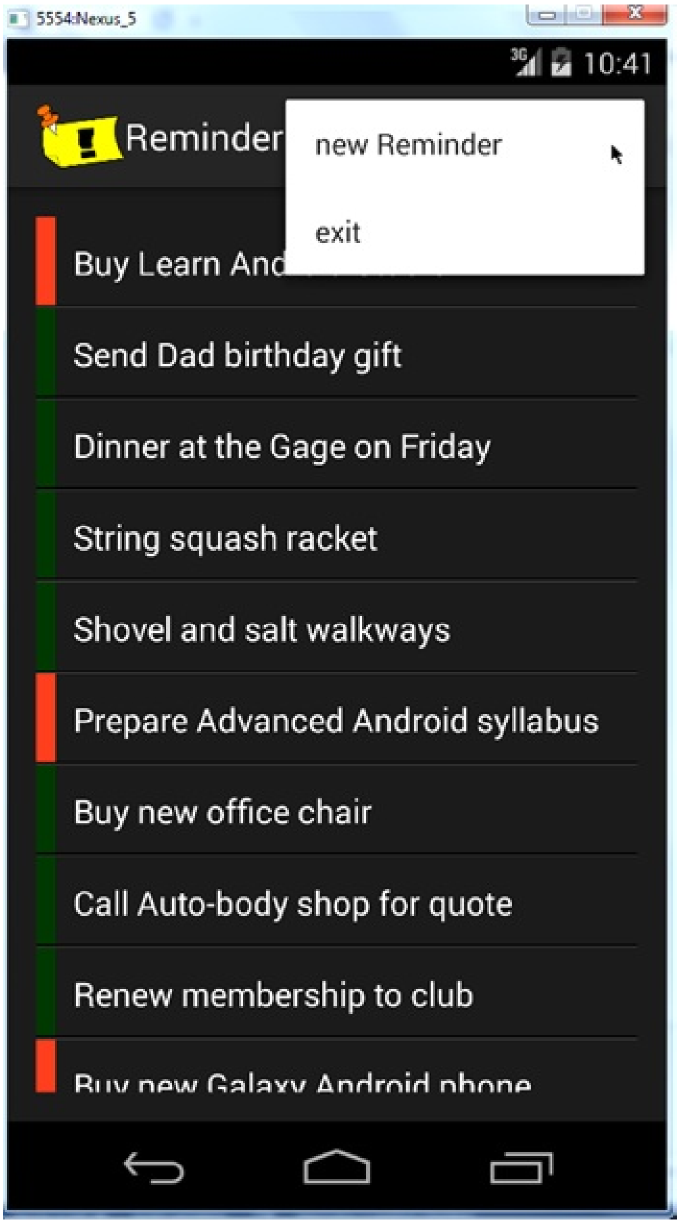

5554:Nexus_5
10:41
Reminder
new Reminder
exit
Buy Learn And
Send Dad birthday gift
Dinner at the Gage on Friday
String squash racket
Shovel and salt walkways
Prepare Advanced Android syllabus
Buy new office chair
Call Auto-body shop for quote
Renew membership to club

图5-4 上下文菜单

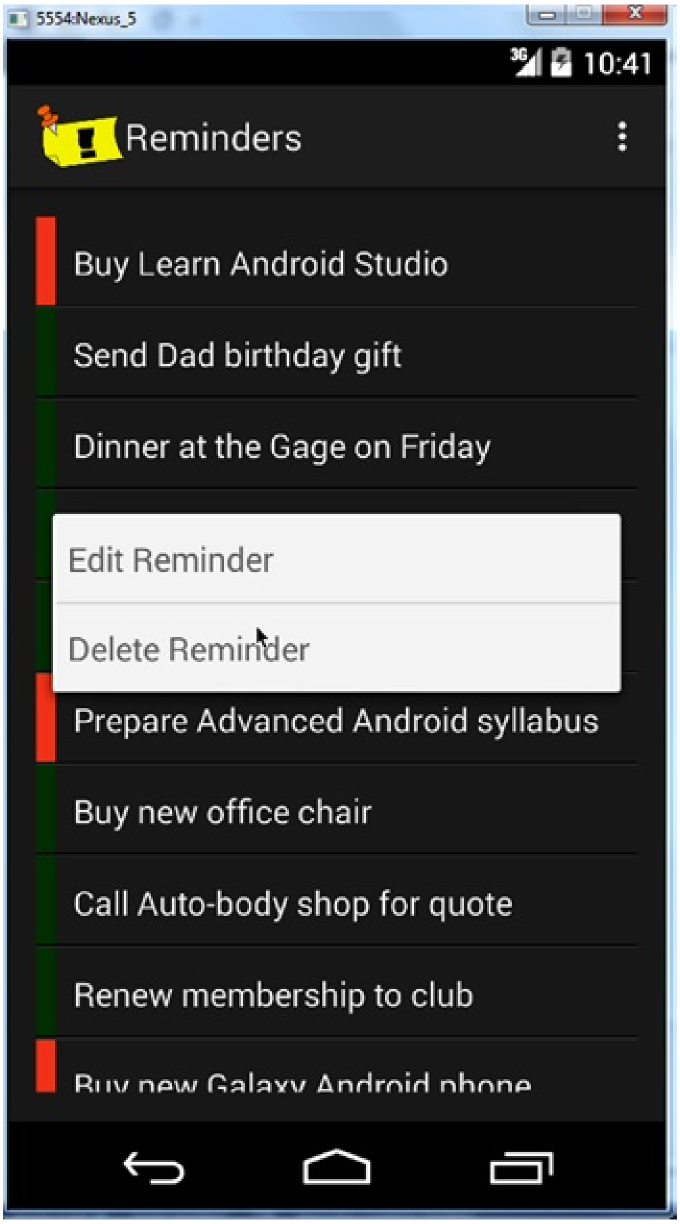


图5-5 编辑提醒对话框

开始一个新项目

在Android Studio开始一个新的项目，用新项目向导如同在第1章里所介绍的。输入Reminders作为项目名，设置公司域名为 `gerber.apress.com` ，并选 `Blank Activity` 模板。保存到 `C:\androidBook\Reminders` 。为了我们例子的一致性，这是一个好主意保存你所有的实验项目到同一个目录，比如 `C:\androidBook` (or use `~/androidBook` 对于 Mac/Linux来说)。在向导的下一页，选择电话和掌上设备并设置最低支持SDK到API 8:Android 2.2(冻酸奶)。通过设定最低支持SDK到API 8，你让这个app支持目前市场上99%的设备。点击下一步按钮，选择 `Blank Activity` ，再点下一步。设置 `activity` 名称为 `RemindersActivity` 并点完成，如图5-6所示。

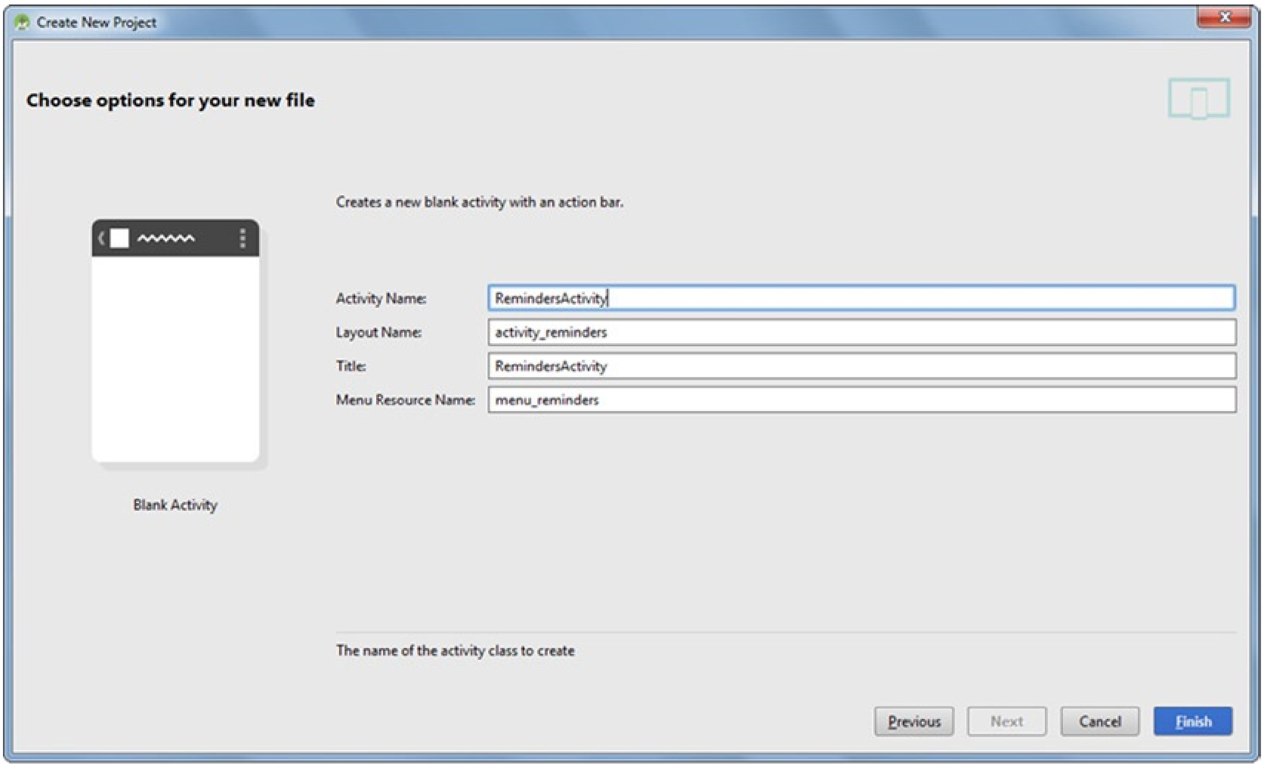


图5-6 输入 `activity` 名称

Android Studio在设计模式下显示 `activity_reminder.xml` 。这是你的主 `activity` 的布局，如图5-7示。如同第1章所讨论的，在这刻项目可以运行在仿真器上或设备上。只要你乐意随便用哪个。

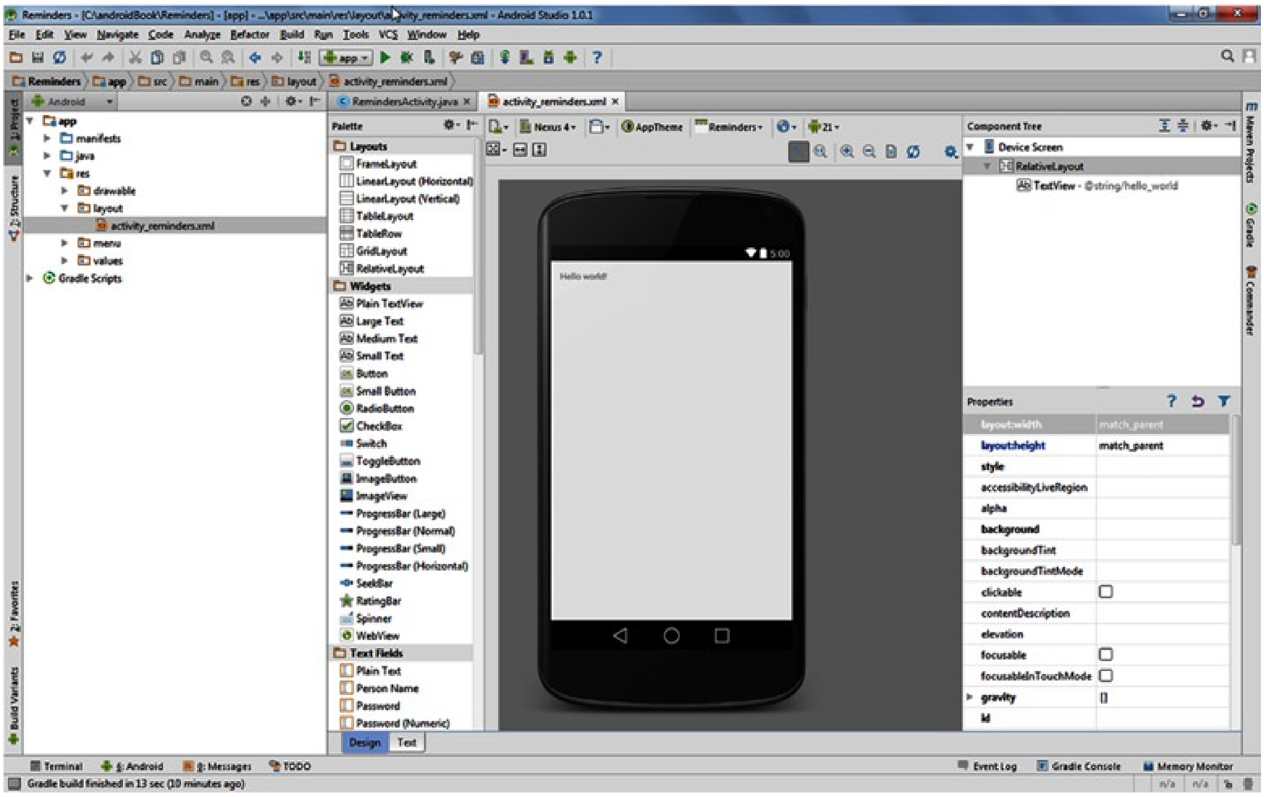


图5-7 `activity_reminders` 的设计模式

初始化Git存储库

建立新项目后的第一步必须是用版本控制来管理源代码。所有这本书的例子都用Git，一个流行的版本控制系统，无缝地协同Android Studio工作并且一直是在线免费的。第7章更彻底地探索了Git和版本控制。

如果你还没安装Git，请参考第7章。从主菜单单选择 `VCS>Import into Version Control>Create Git Repository` 。（在IOS上，选择 `VCS >VCS Operations>Create Git Repository` ）图5-8和5-9展示了这个流程。

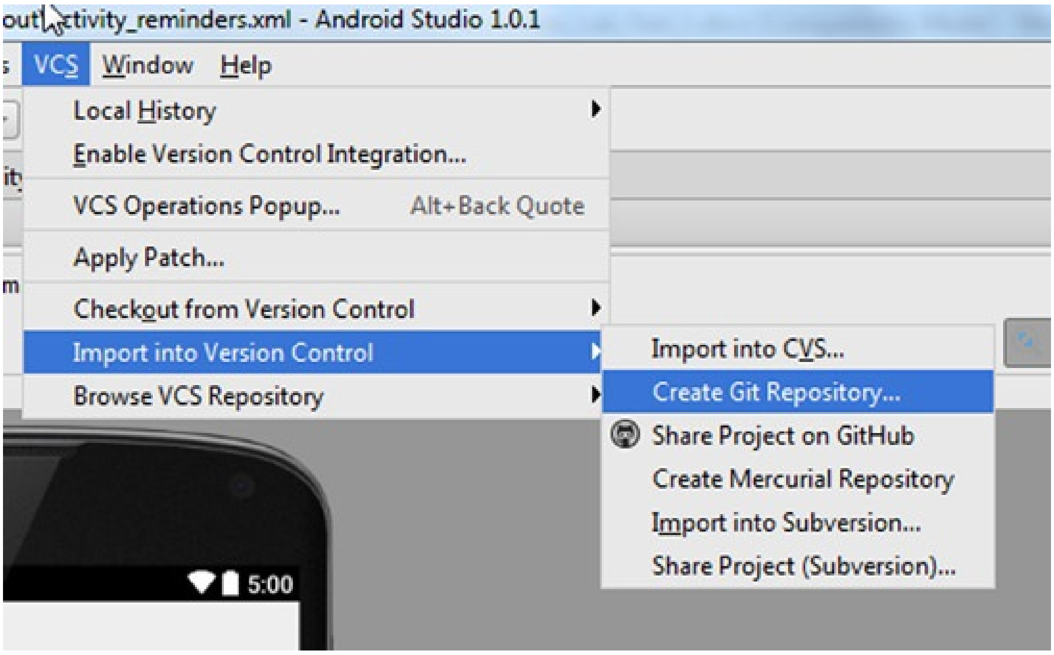


图5-8 创建一个Git存储库

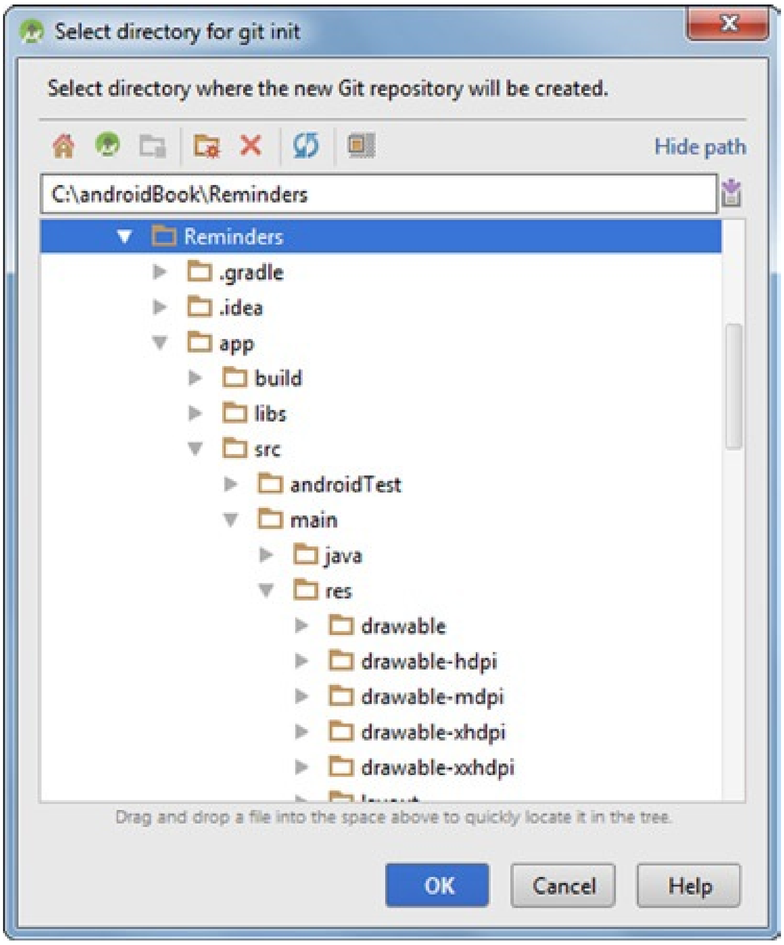


图5-9 为Git存储库选择根目录

当提示选择Git的初始目录，确认初始化为项目的根目录（再次，在这个例子里是Reminders）。点击OK。

你将注意到项目工具窗里大部分文件变成棕色的，意味着它们被Git跟踪但还没加入到Git的存储库而且时刻表也没加载。一但你的项目被Gitr控制，Android Studio使用一个色彩策略，随着我们项目的进行将会解释更多的细节。当然你也可能在这： jetbrains.com/idea/help/file-status-highlights.html 得更多的细节，如果你想研究的话。

点击位于窗口底部边缘的Changes工具按钮切换打开Changes工具窗口并展开叶子标签的未受版本控制文件。这将显示所有被跟踪的文件。为加载它们，选择未受版本控制文件叶子并按 `Ctrl+Alt+A | Cmd+Alt+A` 或右击未受版本控制文件叶子并选择 `Git>Add` 。棕色文件将会变成绿色，意味着它们在Git中已阶段化而且现在准备被提交了。

按 `Ctrl+K | Cmd+K` 来调用提交更改对话框。提交文件是Git版本控制系统记录项目更改的一个过程。如图5-10所示，授权者下拉菜单用于重写当前缺省提交者。让这栏空着吧，这样Android Studio将简单地用你安装Git时设的缺省值。去选Before Commit多选框里所有的选项。把下面的信息放入到Commit Message区里：Initial commit using new project wizard。点击提交按钮并在下拉条目里再次选择提交。

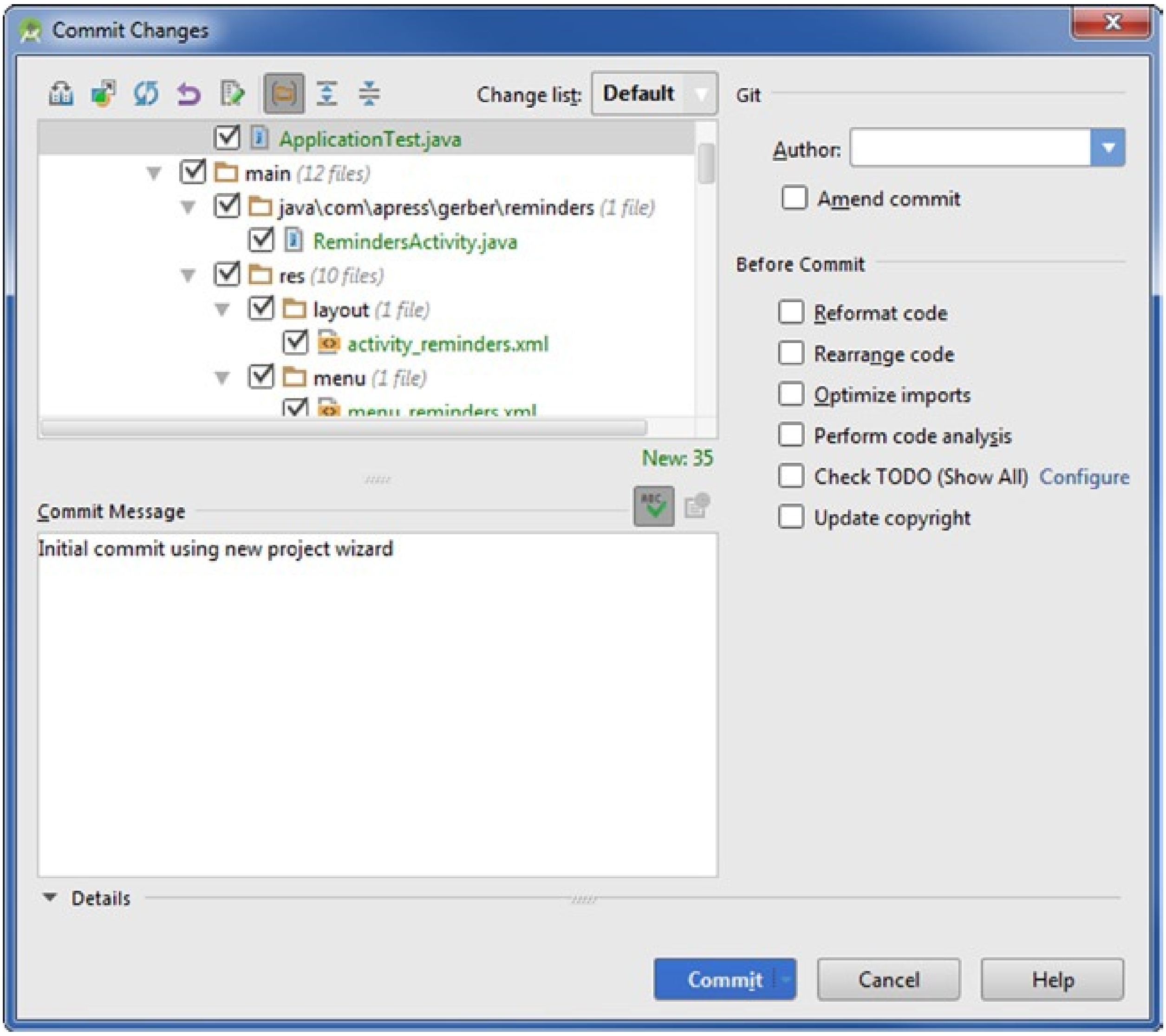


图5-10 提交更改到Git

默认情况下，项目工具窗将会打开。项目工具窗以不同的方式组织你的项目，取决于顶部窗口的下拉菜单所选择的示图。缺省地，下拉菜单是Android示图，它按文件目的组织文件而不是按你电脑操作系统组织文件的方式。当你展开项目工具窗，将注意到三个文件夹在app文件夹下：manifests, java, 和 res。Manifests文件夹里有你的Android manifest文件。Java文件夹是存放java 源文件的地方。Res文件夹保存所有你的Android资源文件。在res目录下的资源可能是XML文件，图象，声音，和其他资源用于帮助定义你的app外观和用户体验。一但你有机会展开Android示图，我们推荐切换到Project示图，它更直观因为它直接反映了你电脑上的目录结构。

构建用户界面

默认情况下，Android Studio打开与主 `activity` 相关联的XML布局文件在一个新的编辑选项卡里并设置为设计模式，因而通常你在新项目里先看到的是可视化开发器。可视化开发器让你编辑app的可视化布局。在屏幕的中央是预览面板。预览面板是Android设备渲染你当前编辑的布局结果的虚拟展示。这个展示可控于屏幕上方横跨的布局预览控制。这些控制可调整

预览并选择不同风格的Android设备，从智能电话到穿戴设备。你也可以改变布局里所描述的相关主题。屏幕的左边，你会发现控件板。包含了众多的控件和 `widget` ，它们可被拖放到正虚拟展示的设备平台上。IDE的右侧包含了一个组件树展示了布局里组件的层次关系。布局使用XML文件。当你在这个可视化开发器作出修改时，这些修改将更新于XML中。你可选择Design或Text选项卡来切换可视化或是文本编辑模式。图5-11标识几个可视化开发器关键区域。

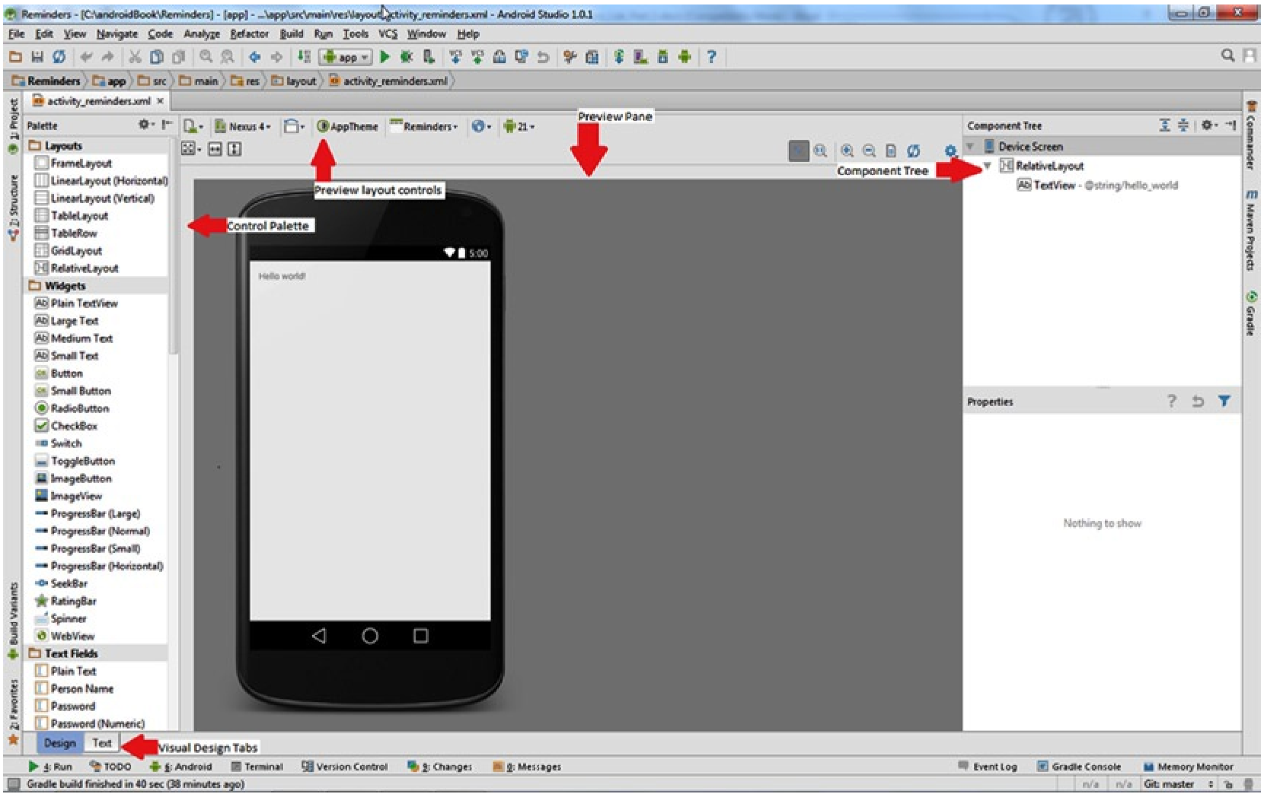


图5-11 可视化开发器界面

工作于图形化编辑器

让我们开始创建reminders的列表项吧。在平台上点击 `Hello World` 文本框并删除它。找到 `ListView` 控件并拖放到平台里。当你拖动时，IDE将显示变化着的尺寸度量以及排列参考来帮助你定们控件，那些当你拖动靠近它们时会企图抓住边缘。拖放 `listView` 它会在屏幕顶部排列。你可以定位在顶部的左边或中间。在定位后，找到在编辑器右下侧的属性示图。设置 `id` 属性为 `reminders_list_view` 。 `Id` 属性是你可以在JAVA代码里控件编程参考；且这个是我们将如何参考的ListView在之后修改代码时。修改 `layout:width` 属性设置为 `match_parent` 。这将扩展控件动态地占有尽可能多的父控件宽度。关于布局在第8章里你会学到更多细节。现在，你的布局将装配成图5-12那样。

图5-12 有个 `ListView` 的 `activity_reminders` 的布局

在Android里，一个`activity`定义了控制用户和app交互的逻辑。当第一次学Android时，把`activity`想象成你的app的一个屏幕是对你有帮助的，虽然`activity`要比这个复杂得多。布局文件定义为一个XML，但用早先提到的图形化编辑器可以图形化编辑。

编辑布局的原生XML

点击编辑器左下方的文本选项卡，从图形化编辑切换到文本编辑模式。这带来了布局的原生XML示图，右边伴随着一个预览面板。将`RelativeLayout`的背景色改为黑色，插入这行：`android:background="#181818"`到`android:layout_height="match_parent"`下面。颜色用十六进制来表达。可以看下第9章关于十六进制颜色的更多信息。注意到你插入那条设置根示图背景色那行后夹缝里的黑灰色的样本。如果你回到设计模式，会观察到整个布局变成黑灰色了。

直接用硬编码一个颜色值到你的XML文件里不是最好的途径。更好的选择是定义一个`colors.xml`文件到资源文件夹里，并在那定义你的颜色。在XML文件（如`colors.xml`）里实现值的原因是那些资源被保存和编辑在一个地方并且在在整个项目里可以轻松地参考。

用`Ctrl+X | Cmd+X`或通过主菜单的`Edit ➤ Cut`选择十六进制的颜色值`#181818`并剪切它到粘贴板。输入`@color/dark_grey`到这个位置。这是一个特殊的语法参考到Android资源文件`colors.xml`，但这个文件还不存在于你的项目里，Android Studio会高亮红色指出这个错误。按`Alt+Enter`会提示纠正这个错误的选项。选择第二个选项，创建颜色资源`dark_grey`，接着把刚才的颜色值粘贴到资源值里：下个出现的对话框值域并点击OK。

新颜色值资源对话框将创建Android资源文件`colors.xml`并填充十六进制的值。点击OK接着在加入文件到Git对话框里还是点OK，这个新文件加入到版本控制，并确保选择Remember,Don’t Ask Again这个复选框因而这个信息下次不会再打扰你了。图5-13演示了这个流程。

图5-13 析出硬代码颜色值到一个资源文件

在预览模式下这个`ListView`控件包含的行布局在我们所选的背景色时没有足够的对比度。要改变这项表象，你将在另外一个布局文件里定义该布局。右击资源文件夹里的布局文件夹并选择`New ➤ Layout Resource File`。在新资源文件对话框里输入`reminders_row`。在根示图组里用线性布局并让其他的为默认值，如图5-14所示。

图5-14 新资源文件对话框

你现在将创建列表项独立的布局。线性布局的根示图组在布局里是最外层的元素。在预览面板顶部的工具栏控制钮设置其走向为垂直方向。当你用这个控制钮时小心点，因为水平线条表示的是垂直方向，反之亦然。图5-15高亮了这个走向按钮。

图5-15 修改走向按钮

在预览面板的右下方找到属性示图。找到`layout:height`属性并设为`50dp`。这个控制控件的高度，`dp`这个单位参照像素独立密度度量系统。这是公开的，Android允许布局参照屏幕重绘时的密度合适地缩放。你可以点击这个示图里的任意属性并增量键入搜索属性，并按上下键继续搜索。

放一个水平线性布局到垂直线性布局里。拖放一个自定义控件到水平线性控件里并设置它的类属性为`android.view.View`，来创建一个通用的空示图并付予`id`为`row_tab`。在写这篇文章时，Android Studio还是有点局限：它不允许你从控件板里拖放通用示图。当你点击`CustomView`时，将得到一个有不同选项的对话框，没有一个包括了通用的`View`类。选择任意一个类并放到你的布局里。为这个限制的情况找到类属性并修改为`android.view.View`。对照清单5-1看看如何完成它

你将会用到这个通用的`View`选项卡来标记那些提醒条是很重要的。还在文本编辑模式时，修改你的用户自定义示图`的layout:width`属性为`10dp`，`layout:height`属性为`match_paren`t。这里用的`match_parent`值将让这个示图控件象它的父控件一样高。切换到设计模式并拖放一个大文本控件到水平线性布局的元件树上，并设置宽高属性为`match_parent`。检查下这个大文本元件定位于用户自定义示图控件的右侧。在元件树里，标志文本视图的元件一定是嵌套在水平线性布局元件里面并且在示图元件下面。如果文本视图在示图元件上面，把它拖下来放到第二（即最后）的位置。给这个文本视图一个id值row_text并设置文字尺寸为`18sp`。`sp`这个单位参照像素独立密度度量系统，象`dp`那样，但它尊照用户的文本尺寸设置，因而比如：如果用户很难看清并想他的手机上的显示更大的文字，`sp`将尊照这个设置，然而`dp`不会。因此，用`sp`作为文字的尺寸单位总是一个好点子。你将在第8章学到有关屏幕度量的更多内容。

最后，设置文本控件的文本属性为`"Reminder Text"`。切换到文本编辑模式做更多的修改如清单5-1所示。

清单5-1 `reminders_row` 布局XML代码

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:orientation="vertical">
    <LinearLayout
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="48dp">
        <view
        android:layout_width="10dp"
        android:layout_height="match_parent"
        class="android.view.View"
        android:id="@+id/row_tab" />
        <TextView
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:textAppearance="?
        android:attr/textAppearanceLarge"
        android:text="Reminder Text"
        android:id="@+id/row_text"
        android:textSize="18sp" />
    </LinearLayout>
</LinearLayout>
```

现在要创建一些用户颜色了。切换到设计模式。选择元件树线性布局（垂直方向的）的根。设置 `android:background` 属性为 `@color/dark_grey` 来重用之前所定义的颜色。在元件树里选择 `row_tab` 元件并设置它的 `android:background` 属性为 `@color/green` 。选择 `row_text` 元件并设置它的 `android:textColor` 属性为 `@color/white` 。在这之前，这些颜色并没设置在 `colors.xml` 里，象之前那样定义它们。切换到到文本模式。按F2在这两个错误间重复地跳前和跳后并按 `Alt_Enter` 带出智能建议。选择第二个建议并在弹出对话框里用 `#ffffff` 和 `#003300` 修正白色和绿色的问题。修正这些问题后，你可以按 `Ctrl` 键和左键点击这些颜色将带到 `colors.xml` 文件里，如清单5-2代码所示。

Listing 5-2. The `colors.xml` File

```
<resources>
    <color name="dark_grey">#181818</color>
    <color name="white">#ffffff</color>
    <color name="green">#003300</color>
</resources>
```

回到 `activity_reminders.xml` 布局文件。现连结新的 `reminders_row` 布局到这个布局了。切换到文本模式并加入现面的属性到 `ListView` 元素里 `tools:listitem="@layout/reminders_row"` ，如图5-16所示。

加上这个属性在运行时并没改变布局的表象；它只不过改变这个 `list view` 的每个条目的预览面板。要让这个新布局有用，你必须用JAVA代码充实它，这个我们将在后续的章节里展示给你。

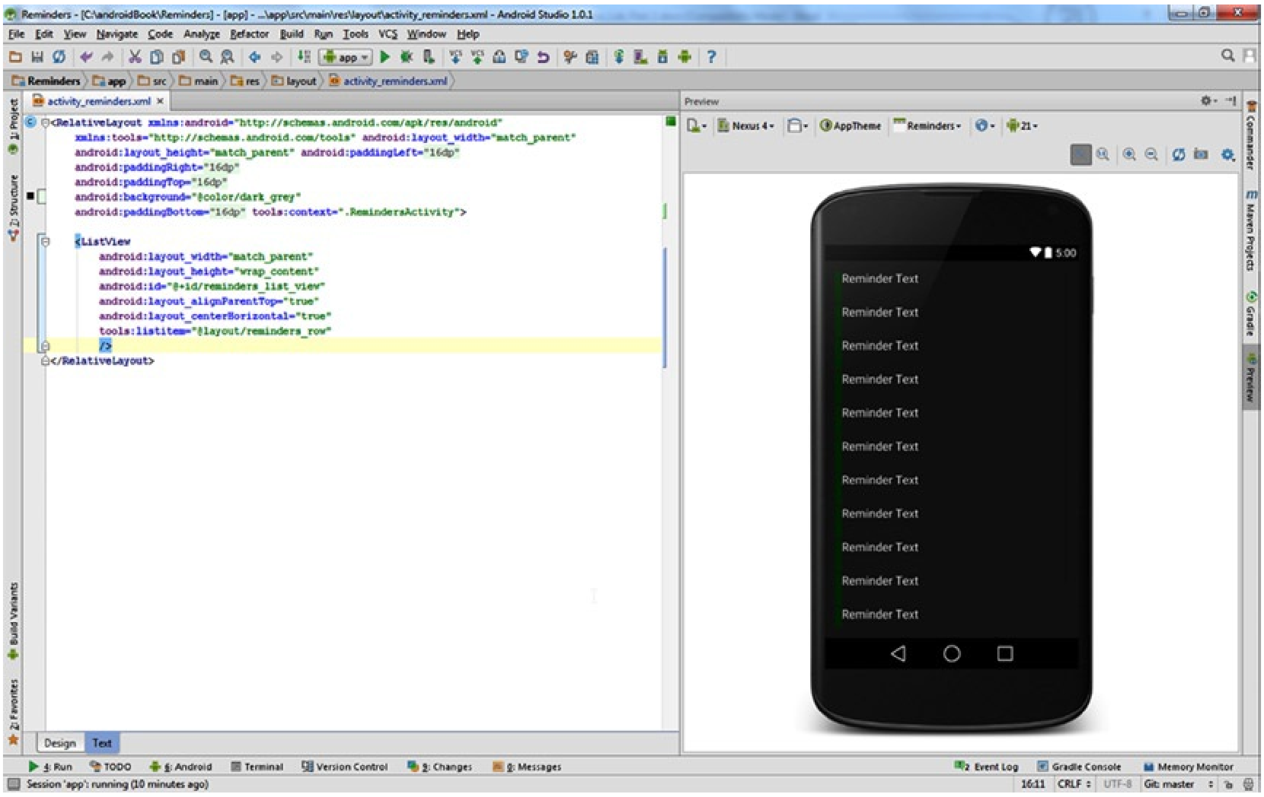


图5-16 预览面板现正展示一个用户自定义的 `ListView` 布局

加入图形增强

刚刚完成一个 `ListView` 条目的用户自定义布局，但你不会停止在这里。加入一些视觉增强元素将让你的app与众不同。看一下文本如何显示在屏幕上。精明点的眼球会看到它有点点偏离中心并跑出绿色选项卡的左边。打开 `reminder_row` 布局做点小调整。你希望文本重心朝行内垂直方向的中心去点，并给出点边衬这样和边界就有点视觉上的分离。用清单5-3的代码代替你的 `TextView` 元素。

Listing 5-3. TextView Additional Attributes

```
<TextView
android:layout_width="match_parent"
android:layout_height="50dp"
android:text="Reminder Text"
android:id="@+id/row_text"
android:textColor="@color/white"
android:textSize="18sp"
android:gravity="center_vertical"
android:padding="10dp"
android:ellipsize="end"
android:maxLines="1"
/>
```

增加的省略属性将截去过长的文本以适应该条目以一个省略号结束，然而 `maxLines` 属性限制了每个条目的行数为1。最后，在内部的线性布局后面但外部线布局结束之前从清单5-4里加上两个更普通的示图对象，来创建条目下的水平规则。外部的线性布局设置高为 `50dp` ，内部

的则设置为 `48dp` 。这两个通用的view对象将占用剩下的垂直的 `2dp` 空间来建造一个斜边。清单5-4如下。

Listing 5-4. Extra Generic Views for beveled edge

```
<LinearLayout>
    <view
    class="android.view.View"
    android:layout_width="fill_parent"
    android:layout_height="1dp"
    android:background="#000"/>
    <view
    class="android.view.View"
    android:layout_width="fill_parent"
    android:layout_height="1dp"
    android:background="#333"/>
</LinearLayout>

<android.view.View
class="android.view.View"
android:layout_width="fill_parent"
android:layout_height="1dp"
android:background="#000"/>
<android.view.View
class="android.view.View"
android:layout_width="fill_parent"
android:layout_height="1dp"
android:background="#333"/>
```

> 译者注：上面代码中的控件 `view` 会有些问题，无法解析背景色属性。最好改成 `android.view.View`

加载条目到ListView

现在刚修改的布局将改变 `activity` 。打开项目工具窗并在JAVA源代码文件夹下找到 `RemindersActivity` 文件。它位于 `com.apress.gerber.reminders` 包里。找到 `onCreate()` 方法。这是你的类里第一个定义的方法。声明一个 `ListView` 成员变量取名 `mListView` 按清单5-5那样修改 `onCreate()` 方法。你将需要解决引入 `ListView` 和 `ArrayAdapter` 。

Listing 5-5. Add List Items to the ListView

```
public class RemindersActivity extends ActionBarActivity {
    private ListView mListView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_reminders);
        mListView = (ListView) findViewById(R.id.reminders_list_view);
        //The arrayAdatper is the controller in our
        //model-view-controller relationship. (controller)
        ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(
            //context
            this,
            //layout (view)
            R.layout.reminders_row,
            //row (view)
            R.id.row_text,
            //data (model) with bogus data to test our listview
            new String[]{"first record", "second record", "third record"});
        mListView.setAdapter(arrayAdapter);
    }
    //Remainder of the class listing omitted for brevity
}
```

代码里用你早先定义 `id` 属性查找 `ListView` 控件，并且去除了默认的条目分隔器，这样我们早先自定义的带斜边分隔器将会很好地渲染。这些代码也创建了一个带有几个例子条目的适配器。适配器是一种特别的JAVA类定义于Android SDK里，它的功能作为在SQLite数据库（模型）里模型-示图-控制器关系的控制器， `ListView` （示图）,还有适配器（控制器）。适配器绑定模型到示图上并且处理更新和刷新。 `Adapter` 是 `ArrayAdapter` 的超类，用来绑定数组到示图里。在我们的例子里，这个示图是 `ListView` 。 `ArrayAdapter` 的构造函数有三个参数。第一个是上下文对象用来表达当前 `activity` 。适配器也需要知道那个布局和布局里的区域或多个区域用于显示行数据。为满足这个要求，你要传送布局和文本示图条目的 `id` 到布局里。最后一个参数是一个字符串数组用于表里的每个条目。如果你这时运行项目，将看到给到 `ArrayAdapter` 构造器的那些值显示如图5-17。



图5-17 ListView例子

按 `Ctrl+K | Cmd+K` 提交更改到Git并填入Adds ListView with custom colors到提交信息里。当你工作于这个项目里，用提交信息描述每次提交的追加/移除/更改来执行增加提交到Git是一个不错的体验。为将来的合作者及用户保持这个习惯来会让事情变得容易点，来鉴别各次单独的提交以及以后的构建记录。

设置溢出菜单的行为条

Android使用一个共同的可视化元素叫做行为条。行为条用于多数app定位导航和其他选项以让用户执行重要的任务。这时运行这个app，你可能注意到一个菜单图标象三个竖直点的那个。这些点被叫做溢出菜单。点击这个溢出菜单图标会产生只有一个叫 `setting` 菜单项的菜单。这个菜单项放在那是新项目向导模板的一部分并且本质上是一个占位没有任何执行动作。 `RemindersActivity` 装载了 `menu_reminders.xml` 文件，可以在 `res/menu` 文件夹下找到。让我们象清单5-6那样对这个文件做点改变加上一些新的菜单项。

Listing 5-6. New Menu Items

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
tools:context="com.apress.gerber.reminders.app.RemindersActivity" >
    <item android:id="@+id/action_new"
    android:title="new Reminder"
    android:orderInCategory="100"
    app:showAsAction="never" />
    <item android:id="@+id/action_exit"
    android:title="exit"
    android:orderInCategory="200"
    app:showAsAction="never" />
</menu>
```

在前述的代码清单里， `title` 属性关联于菜单项的显示文本。因为我们用硬代码设置这些属性值，Andorid Studio将会标记警告。按F2在这些警告间跳跃并按 `Alt+Enter` 拉出智能建议。你只需简单地按回车接受第一个建议，输入新字符串资源的名称，接着马上弹出对话框，再按回车接受命名的资源。用 `new_reminder` 作为第一个条目的名称，第二个叫 `exit` 。

打开 `RemindersActivity` 并用清单5-7的代码代替 `onOptionItemSelected()` 方法。你还需要解决 `Log` 类的引入。当你点击app上的一个菜单项时，实时调用这个方法，传入那个被点击的菜单项的引用。 `Switch` 语句用 `MenuItem` 的 `itemId` ，执行一个 `log` 语句或终结这个 `activity` ，取决于哪个菜单项被点击了。这个例子用 `Log.d()` 方法写入文本到Android的调试日志里。如果你的app包括多个 `activity` 并且这些 `activity` 比当前 `activity` 先显示，那么调用 `finish()` 将简单地把当前 `activity` 推出栈并且把控制交给下面的 `activity` 。因为 `RemindersActivity` 是这个app的唯一 `activity` ， `finish()` 方法推出它将导致这个app的终止。

Listing 5-7. `onOptionsItemSelected( )` Method Definition

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_new:
            //create new Reminder
            Log.d(getLocalClassName(),"create new Reminder");
            return true;
        case R.id.action_exit:
            finish();
            return true;
        default:
            return false;
    }
}
```

运行这个app并测试新的菜单选项。点击new Reminder菜单项并观察Android日志里出现的信息。Android DDMS（Dalvik调试监控服务）窗将与仿真器或设备运行app的同时打开，并且你将需要选择调试选项低于Log级别才可看到调试日志。运行你的app并与这些菜单项互动。

注意在你点击New Reminder菜单项时Android DDMS出来的日志。最后，按`Ctrl+K | Cmd+K`并提交你的代码到Git，用Adds new reminder and exit menu options作为提交信息。

保持提醒

因为Reminders这个app需要保存提醒清单，你将需要一个保存策略。Android SDK和运行时系统提供了一个内建的数据库引擎叫SQLite，它设计为有限记忆体环境并很适合于移动设备。这节涵括了SQLite数据库并浏览怎么保存提醒清单。我们的策略将包括一个数据模型，一个数据库代理类，还有一个游标适配器(`CursorAdapter`)。这个模型将维持数据读写到数据库。代理将是一个适配器类将把从app简单调用转换为到数据库API调用。最后，游标适配器将扩展为一个标准的Android类用于以抽象方式处理数据访问。

数据模型

让我们开始创建数据模型。右击`com.apress.gerber.reminders`包并选择`New ➤ Java Class`。命名这个类为`Reminder`并按回车。如清单5-8那样装裱这个类。这是一个简单的POJO（简单的老JAVA对象）定义了几个实例变量和相应的`getter`及`setter`方法。`Remider`类包含了和个整型的`ID`，字符串变量，和数值化的重要值。`ID`是用于标记每个提醒的唯一数字。字符串则保存了提醒的文本。重要值是一个数值化的指示器用来标志一个独立的提醒是否重要(1=重要，0=不重要)。我们更愿意用整型而不是布尔值是因为SQLite数据库没有布尔数据类型。

Listing 5-8. `Reminder` Class Definition

```
public class Reminder {
    private int mId;
    private String mContent;
    private int mImportant;
    public Reminder(int id, String content, int important) {
        mId = id;
    mImportant = important;
        mContent = content;
    }
    public int getId() {
        return mId;
    }
    public void setId(int id) {
        mId = id;
    }
    public int getImportant() {
        return mImportant;
    }
    public void setImportant(int important) {
        mImportant = important;
    }
    public String getContent() {
        return mContent;
    }
    public void setContent(String content) {
        mContent = content;
    }
}
```

现在你将创建一个数据代理。再次，这个代理将转换简单的应用调用为低级别的SQLite API调用。在 `com.apress.gerber.reminders` 包里新建一个类叫 `RemindersDbAdapter` 。把清单5-9的代码直接加入到这个新类里。当你解析导入时，发现D `atabaseHelper` 并没在Android SDK里。我们将在后面的步骤里定义 `DatabaseHelper` 。这些代码定义了列名称和索引；一个TAG作日志；两个上下文对外对象；和一个SQL语句用于创建数据库。

Listing 5-9. Code to be placed inside the `RemindersDbAdapter` class

```
//these are the column names
public static final String COL_ID = "_id";
public static final String COL_CONTENT = "content";
public static final String COL_IMPORTANT = "important";
//these are the corresponding indices
public static final int INDEX_ID = 0;
public static final int INDEX_CONTENT = INDEX_ID + 1;
public static final int INDEX_IMPORTANT = INDEX_ID + 2;
//used for logging
private static final String TAG = "RemindersDbAdapter";
private DatabaseHelper mDbHelper;
private SQLiteDatabase mDb;
private static final String DATABASE_NAME = "dba_remdrs";
private static final String TABLE_NAME = "tbl_remdrs";
private static final int DATABASE_VERSION = 1;
private final Context mCtx;
//SQL statement used to create the database
private static final String DATABASE_CREATE =
    "CREATE TABLE if not exists " + TABLE_NAME + " ( " +
    COL_ID + " INTEGER PRIMARY KEY autoincrement, " +
    COL_CONTENT + " TEXT, " +
    COL_IMPORTANT + " INTEGER );";
```

SQLite API

DatabaseHelper是一个SQLite API类，用于打开和关闭数据库。它用到了上下文 `Context` ，这是一个抽象的Android类以提供到Android操作系统的访问。 `DatabaseHelper` 是一个用户自定义类，且必须由你来定义。用清单5-10实施 `DatabaseHelper` 类，作为 `RemindersDbAdapter` 的内部类。把这些代码放到 `RemindersDbAdapter` 后段，但在类结束的花括号的前面。

Listing 5-10. `RemindersDbAdapter`

```
private static class DatabaseHelper extends SQLiteOpenHelper {
    DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        Log.w(TAG, DATABASE_CREATE);
        db.execSQL(DATABASE_CREATE);
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        Log.w(TAG, "Upgrading database from version " + oldVersion + " to "
            + newVersion + ", which will destroy all old data");
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
        onCreate(db);
    }
}
```

`DatabaseHelper` 继承于 `QLiteOpenHelper` ，用于以特殊的回调方法来维护数据库。回调方法是运行时环境在应用的整个生命周期内都调用的方法，并且它们用到 `SQLiteDatabase db` 变量提供SQL命令的执行。在构造器里数据库就被初始化了。构造器传递数据库名和版本到它的父类；然后父类做数据库的设置繁杂工作。当需要建立数据库时 `onCreate()` 方法被运行时自动调用。这个操作只运行一次，当app首次启动并且数据库还未创建时。 `onUpgrade()` 方法是在数据库更新时被调用，例如假如开发者改变纲要。如果你改变数据库的纲要，确保给 `DATABASE_VERSION` 增加1，然后 `onUpgrade()` 会处理剩下的事情。如果你忘了给 `DATABASE_VERSION` 加1，你的app将崩溃即便在调试构建模式下。在前述的代码里，在运行 `onCreate()` 方法之前我们运行一个SQL命令来移除数据库里唯一的表单，以方便重建表单。

清单5-11演示了用 `DatabaseHelper` 来打开和关闭数据库。构造器保存了上下文的一个实例，它会被传送到 `DatabaseHelper` 。 `open( `)方法初始化helper并用它得到一个数据库的实例，直到 `close()` 方法用于关闭数据库。把这些代码加入到 `RemindersDbAdapter` 类里，在所有成员变量之后和 `DatabaseHelper` 内部类之前。当你解析导入时，用 `android.database.SQLException` 类。

Listing 5-11. Database Open and Close Methods

```
public RemindersDbAdapter(Context ctx) {
    this.mCtx = ctx;
}
//open
public void open() throws SQLException {
    mDbHelper = new DatabaseHelper(mCtx);
    mDb = mDbHelper.getWritableDatabase();
}
//close
public void close() {
    if (mDbHelper != null) {
        mDbHelper.close();
    }
}
```

清单5-12包含了所有处理在tbl_remdrs表单里创建，读取，更新，和删除 `Reminder` 对象的操作。通常参考CRUD操作；CRUD即代表创建，读取，更新，删除。加入这些代码到 `RemindersDbAdapter` 类的 `close()` 方法之后。

Listing 5-12. Database CRUD Operations

```
//CREATE
//note that the id will be created for you automatically
public void createReminder(String name, boolean important) {
    ContentValues values = new ContentValues();
    values.put(COL_CONTENT, name);
    values.put(COL_IMPORTANT, important ? 1 : 0);
    mDb.insert(TABLE_NAME, null, values);
}
//overloaded to take a reminder
public long createReminder(Reminder reminder) {
    ContentValues values = new ContentValues();
    values.put(COL_CONTENT, reminder.getContent()); // Contact Name
    values.put(COL_IMPORTANT, reminder.getImportant()); // Contact Phone Number
    // Inserting Row
    return mDb.insert(TABLE_NAME, null, values);
}
//READ
public Reminder fetchReminderById(int id) {
    Cursor cursor = mDb.query(TABLE_NAME, new String[]{COL_ID,
        COL_CONTENT, COL_IMPORTANT}, COL_ID + "=?",
        new String[]{String.valueOf(id)}, null, null, null, null
    );
    if (cursor != null)
        cursor.moveToFirst();
    return new Reminder(
        cursor.getInt(INDEX_ID),
        cursor.getString(INDEX_CONTENT),
        cursor.getInt(INDEX_IMPORTANT)
    );
}
public Cursor fetchAllReminders() {
    Cursor mCursor = mDb.query(TABLE_NAME, new String[]{COL_ID,
        COL_CONTENT, COL_IMPORTANT},
        null, null, null, null, null
    );
    if (mCursor != null) {
        mCursor.moveToFirst();
    }
    return mCursor;
}
//UPDATE
public void updateReminder(Reminder reminder) {
    ContentValues values = new ContentValues();
    values.put(COL_CONTENT, reminder.getContent());
    values.put(COL_IMPORTANT, reminder.getImportant());
    mDb.update(TABLE_NAME, values,
        COL_ID + "=?", new String[]{String.valueOf(reminder.getId())});
}
//DELETE
public void deleteReminderById(int nId) {
    mDb.delete(TABLE_NAME, COL_ID + "=?", new String[]{String.valueOf(nId)});
}
public void deleteAllReminders() {
    mDb.delete(TABLE_NAME, null, null);
}
```

每个方法都用到了 `SQLiteDatabase mDb` 变量来生成和执行SQL语句。如果你熟悉SQL，你会猜到这些SQL语句将由 `INSERT` ， `SELECT` ， `UPDATE` ，或 `DELETE` 组成。

两个创建方法用到了特别的 `ContentValues` 对象，这个是数据载体用于在 `insert` 方法里传送数据值到数据库对象。在 `insert` 语句里数据库最终将转换这些对象到数据库。有两个读方法，一个用于引出单个提醒另一个则引出一个游标以遍历所有的提醒。晚点在一个特别的 `Adapter` 类里你将用到游标。

更新方法象第二个创建方法。无论怎样，这个方法调用了底层的数据库对象的更新方法，那个将生成并执行一个更新SQL语句而不是一个插入语句。

最后，有两个删除方法。第一个针对特定的提醒用 `id` 参数和数据库对象来生成和执行一条删除语句。第二个方法需要数据库生成并执行一条删除语句来移除所有表单里的提醒。

最后，有两个删除方法。第一个针对特定的提醒用 `id` 参数和数据库对象来生成和执行一条删除语句。第二个方法需要数据库生成并执行一条删除语句来移除所有表单里的提醒。

这时，你需要一个手段从数据库提出提醒并放入到 `ListView` 。清单5-13演示了必要的逻辑，通过继承之前你看到的特别的 `Adapter` Android类来绑定数据库值到单独的行对象。创建一个新类 `RemindersSimpleCursorAdapter` 在 `com.apress.gerber.reminders` 包下，并完成处理代码。当解析导入类时，使用 `android.support.v4.widget.SimpleCursorAdapter` 类。

Listing 5-13. R `emindersSimpleCursorAdapter` Code

```
public class RemindersSimpleCursorAdapter extends SimpleCursorAdapter {
    public RemindersSimpleCursorAdapter(Context context, int layout, Cursor c, String[
] from, int[] to, int flags) {
        super(context, layout, c, from, to, flags);
    }
    //to use a viewholder, you must override the following two methods and define a Vi
ewHolder class
    @Override
    public View newView(Context context, Cursor cursor, ViewGroup parent) {
        return super.newView(context, cursor, parent);
    }
    @Override
    public void bindView(View view, Context context, Cursor cursor) {
        super.bindView(view, context, cursor);
        ViewHolder holder = (ViewHolder) view.getTag();
        if (holder == null) {
            holder = new ViewHolder();
            holder.colImp = cursor.getColumnIndexOrThrow(RemindersDbAdapter.COL_IMPORT
ANT);
            holder.listTab = view.findViewById(R.id.row_tab);
            view.setTag(holder);
        }
        if (cursor.getInt(holder.colImp) > 0) {
            holder.listTab.setBackgroundColor(context.getResources().getColor(R.color.
orange));
        } else {
            holder.listTab.setBackgroundColor(context.getResources().getColor(R.color.
green));
        }
    }
    static class ViewHolder {
        //store the column index
        int colImp;
        //store the view
        View listTab;
    }
}
```

我们用适配器把所有提醒登记到 `ListView` 。在运行时中， `ListView` 将重复调用在适配器里的 `bindView()` 方法，以单独的示图对象作为用户装载器，且在清单里滚动。填入这些清单条目到示图里是适配器的工作。在这个例子代码里，我们使用了适配器的子类 `SimpleCursorAdapter` 。这个类用了一个游标对象，它保存着跟踪表单里的行轨迹。

这里你看到了一个 `ViewHolder` 模式的例子。这是一个容易认识的Android模式，每个 `ViewHolder` 对象含有一个示图的标签。用数据源（在这个例子里是 `Cursor` ）的值，这个对象加载清单里的示图对象。 `ViewHolder` 定义为一个内部静态类，有两个实例变量，一个用于索引重要的表单项另一个用于在布局里定义的 `row_tab` 示图。

`bindView()` 方法开始于父类在示图里从游标到元素的 `map` 值方法的调用。然后检查看（holder）是否附有一个标签还是有必要创建一个新的（holder）。然后 `bindView()` 方法用重要列索引和早先定义的 `row_tab` 配置容器的实例变量。在容器被发现或配置后，从当前提醒的 `COL_IMPORTANT` 常量秋决定 `row_tab` 用什么颜色。这个例子用了新的橙色，那个你要加到 `colors.xml` ： `<color name="orange">#ffff381a</color>` 。

早先你用了 `ArrayAdapter` 来管理模型和示图之间的关系。 `SimpleCursorAdapter` 也是用同样的模式，虽然它的模型是SQLite数据库。将清单5-14更改到你的 `RemindersDbAdaper` 和 `RemindersSimpleCursorAdapter` 类里。

Listing 5-14. `RemindersActivity` Code

```
public class RemindersActivity extends ActionBarActivity {
    private ListView mListView;
    private RemindersDbAdapter mDbAdapter;
    private RemindersSimpleCursorAdapter mCursorAdapter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_reminders);
        mListView = (ListView) findViewById(R.id.reminders_list_view);
        mListView.setDivider(null);
        mDbAdapter = new RemindersDbAdapter(this);
        mDbAdapter.open();
        Cursor cursor = mDbAdapter.fetchAllReminders();
        //from columns defined in the db
        String[] from = new String[]{
            RemindersDbAdapter.COL_CONTENT
        };
        //to the ids of views in the layout
        int[] to = new int[]{
            R.id.row_text
        };
        mCursorAdapter = new RemindersSimpleCursorAdapter(
            //context
            RemindersActivity.this,
            //the layout of the row
            R.layout.reminders_row,
            //cursor
            cursor,
            //from columns defined in the db
            from,
            //to the ids of views in the layout
            to,
            //flag - not used
        0);
        // the cursorAdapter (controller) is now updating the listView (view)
        //with data from the db (model)
        mListView.setAdapter(mCursorAdapter);
    }
    //Abbreviated for brevity
}
```

这时候如果你运行app，你还是看不到清单里有任何东西；屏幕完全是空的，因为你最后的修改在例子的数据部分插入了数据库功能。按`Ctrl+K | Cmd+K`并提交的修改，提交信息：Adds SQLite database persistence for reminders and a new color for important reminders。聪明点的话，你可能想弄清楚怎么用新的`RemindersDbAdaper`把例子里的条目加回来。这将在下章描述，你可以继续看下去并检查下作业了。

小结

致此，你有了个成熟的Android应用。在这章，你学会了怎样设置第一个Android项目并用Git控制代码。你也探索了怎么编辑Android布局，用设计或文本模式。你也看到建立一个在动作条里的溢出菜单。这章的最后探索了`ListView`和`Adapter`，以及绑定数据到内建的SQLite数据库。在接下来的章节里，增加创建和编辑提醒的功能，你将完成这个app。

第六章：Reminders实验：第二部分

> 译者：qiangxcn
>
> 来源：Learn Android Studio 汉化教程 第六章：Reminders实验：第二部分

这章涵括了通过对话框捕获用户输入。也继续演示适配器及SQLite数据库的使用。这章里，我们将完成从第五章开始的例子。

增加/删除提醒

第五章里这个例子的屏幕还没有任何提醒。为了让布局看到提醒清单，当app启动时加载些提醒的例子上去，这是很有用的。如果你想挑战这章处理过程，比较下清单6-1和你的代码。清单6-1检查是否有保存的实例，如果有，处理将设置例子数据。为此，代码调用了些DatabaseAdapter的方法；一个是清除所有的提醒，另一个则是加入些提醒。

Listing 6-1. Add Some Example Reminders

```
public class RemindersActivity extends ActionBarActivity {
    private ListView mListView;
    private RemindersDbAdapter mDbAdapter;
    private RemindersSimpleCursorAdapter mCursorAdapter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_reminders);
        mListView = (ListView) findViewById(R.id.reminders_list_view);
        mListView.setDivider(null);
        mDbAdapter = new RemindersDbAdapter(this);
        mDbAdapter.open();
        if (savedInstanceState == null) {
            //Clear all data
            mDbAdapter.deleteAllReminders();
            //Add some data
            mDbAdapter.createReminder("Buy Learn Android Studio", true);
            mDbAdapter.createReminder("Send Dad birthday gift", false);
            mDbAdapter.createReminder("Dinner at the Gage on Friday", false);
            mDbAdapter.createReminder("String squash racket", false);
            mDbAdapter.createReminder("Shovel and salt walkways", false);
            mDbAdapter.createReminder("Prepare Advanced Android syllabus", true);
            mDbAdapter.createReminder("Buy new office chair", false);
            mDbAdapter.createReminder("Call Auto-body shop for quote", false);
            mDbAdapter.createReminder("Renew membership to club", false);
            mDbAdapter.createReminder("Buy new Galaxy Android phone", true);
            mDbAdapter.createReminder("Sell old Android phone - auction", false);
            mDbAdapter.createReminder("Buy new paddles for kayaks", false);
            mDbAdapter.createReminder("Call accountant about tax returns", false);
            mDbAdapter.createReminder("Buy 300,000 shares of Google", false);
            mDbAdapter.createReminder("Call the Dalai Lama back", true);
            }
        //Removed remaining method code for brevity...
    }
    //Removed remaining method code for brevity...
}
```

有几个 `createReminder()` 方法的调用，每个都是用一个字符串作提醒文本，以及一个布尔值标记提醒是否重要。我们设置一些实体值让显示好看点。选中 `createReminder()` 方法调用的代码块按以 `Ctrl+Alt+M | Cmd+Alt+M` 汲取方法，如图6-1示。这是一个通过重构菜单和快捷键结合的重构操作。输入 `insertSomeReminders` 作为方法名并确认。这些代码块将以析出的方法代替，这些代码块将在方法体里了。

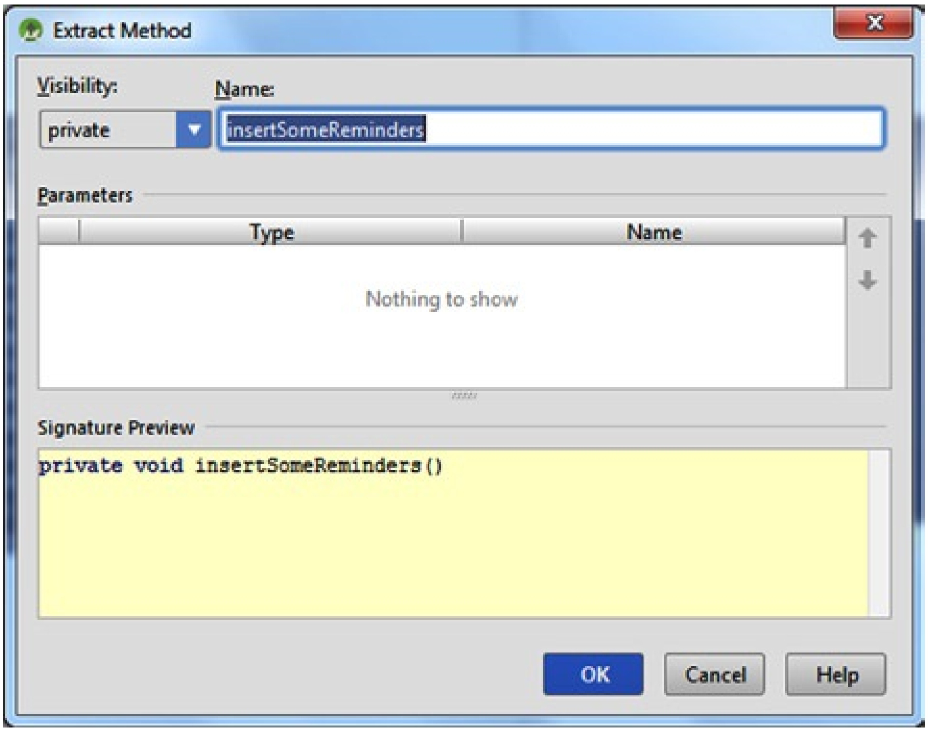


图6-1 析出方法对话框，创建一个 `insertSomeReminders()` 方法

运行app看到的界面，拥有提醒例子了。你的app应该看起来象图6-2的截屏那样。有些提醒显示绿色的行选项卡，而那些重要的提醒则显示橙色行选项卡。提交你的更改到Git，备注Adds Example reminders。

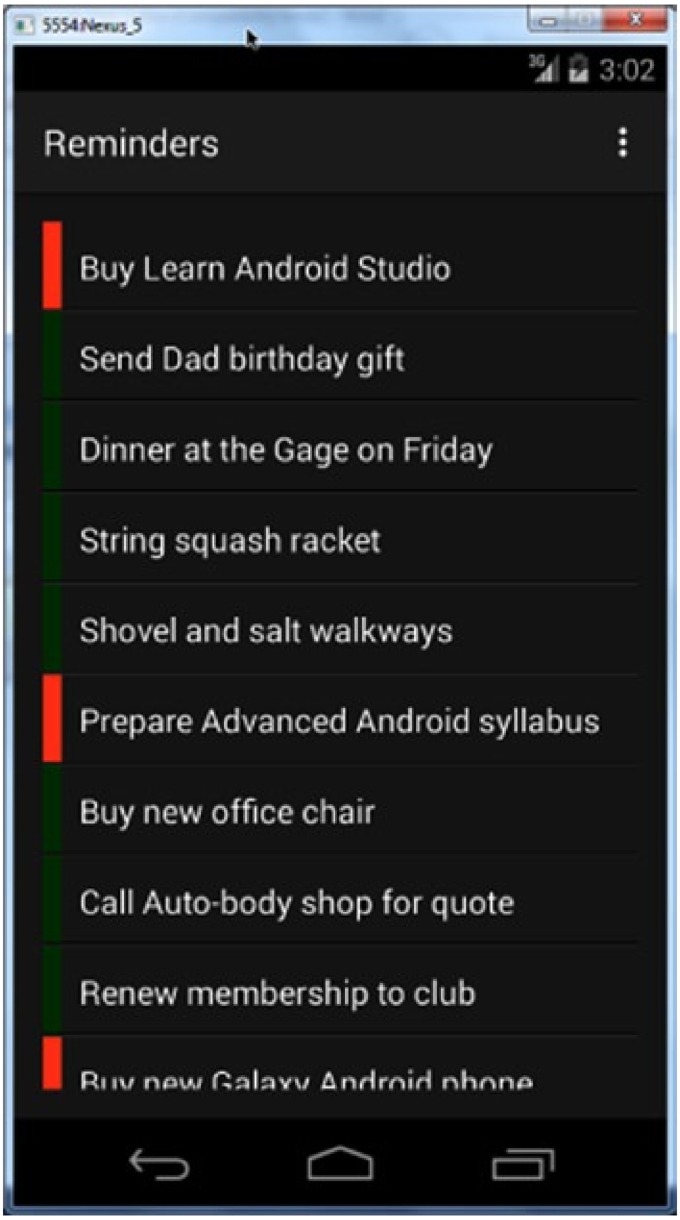


图6-2 插入了提醒例子的实时运行

响应用户的互动

没有app不响应输入。在这节，你将加入响应点击事件的逻辑并且最终允许用户编辑独立的提醒。在app里的主要元素是 `ListView` ，一个Android `View` 的子类。直致此刻，除了把它放到布局里，你还没做过其它的什么。 `android.view.View` 是所有你看到的所有屏幕元素的超类。

把清单6-2的代码加到 `RemindersActivity` 的 `onCreate()` 方法后面，即在方法结束花括号之前，并解决导入类的问题。这是一个匿名内部类实现 `OnItemClickListener` 接口，它只有一个方法， `onItemClicked()` 。这个对象将用于你互动与它所跟踪的 `ListView` 元素的实时运行。当用户点击 `ListView` 时，匿名内部类的 `onCreate()` 方法将被调用。我们定义一个吐司，一个Android SDK的类。调用 `Toast.makeText()` 将导致在屏幕上弹出一个小菜单，显示出你传送给方法的文本。你可以看到清单6-2的代码，作为快速正确使用吐司的指引。

注意：吐司信息可能在一些设备上被隐藏。另一个替代途径是记录一个日志信息，用Android的日志记录器，那个在第十二章会有详述。

Listing 6-2. Set an `OnItemClickListener` with a `Toast`

```
//when we click an individual item in the listview
mListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Toast.makeText(RemindersActivity.this, "clicked " + position,
            Toast.LENGTH_SHORT).show();
    }
});
```

点击屏幕第一个提醒将调用 `onItemClick()` 方法，在清单里的位置是0，所以索引是0。这个逻辑将弹出位置信息的文本，如图6-3所示。

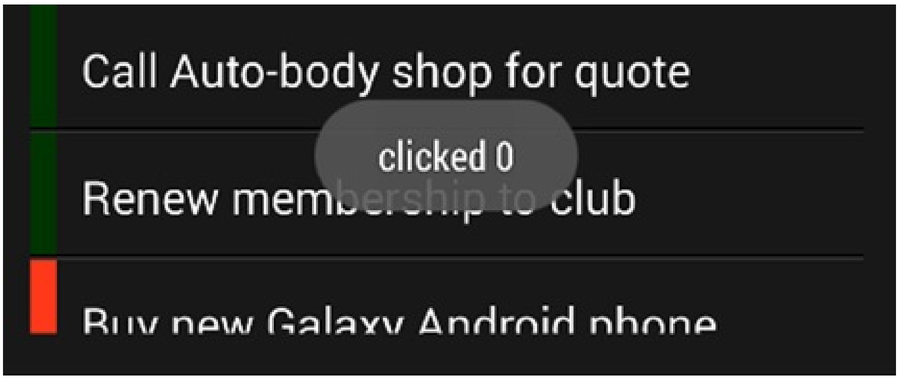


图6-3 点击第一条提醒弹出的吐司信息

用户对话框

大家所熟悉的一些点击事件，现在你可增强点击监听为显示一个对话框。用清单6-3的代码替换所有的 `onItemClick()` 方法。解决导入时，请用 `android.support.v7.app.AlertDialog` 类。

Listing 6-3. `onItemClick( )` Modifications to Allow Edit/Delete

```
public void onItemClick(AdapterView<?> parent, View view, final int masterListPosition
, long id) {
    AlertDialog.Builder builder = new AlertDialog.Builder(RemindersActivity.this);
    ListView modeListView = new ListView(RemindersActivity.this);
    String[] modes = new String[] { "Edit Reminder", "Delete Reminder" };
    ArrayAdapter<String> modeAdapter = new ArrayAdapter<>(RemindersActivity.this,
    android.R.layout.simple_list_item_1, android.R.id.text1, modes);
    modeListView.setAdapter(modeAdapter);
    builder.setView(modeListView);
    final Dialog dialog = builder.create();
    dialog.show();
    modeListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long i
d) {
            //edit reminder
            if (position == 0) {
                Toast.makeText(RemindersActivity.this, "edit "
                    + masterListPosition, Toast.LENGTH_SHORT).show();
                //delete reminder
            } else {
                Toast.makeText(RemindersActivity.this, "delete "
                    + masterListPosition, Toast.LENGTH_SHORT).show();
            }
            dialog.dismiss();
        }
    });
}
```

在处理代码里你看到了另Android类在工作， `AlertDialog.Builder` 。这个类创建器嵌入在 `AlertDialog` 类里，是个静态类，它用于创建 `AlertDialog` 。

在这个实例里的代码远超于创建一个 `ListView` 以及用 `ArrayAdapter` 喂入条目到 `ListVite` 。你可能回想起第五章的这种模型。这个章节建立一个有两个潜在元素（选项）的数组，编辑提醒和删除提醒，在传送到 `ListView` 之前，这个是，轮流地，传送给 `AlertDialog.Builder` 。这个创建器然后用这些选项清单创建并显示一个对话框。

注意下代码清单6-3最后的选择部分。有点象之前的 `OnItemClickListener()` ；不管怎样，我们附加了一个 `modeListView` 的监听器，那个在当前 `OnItemClickListener` 里创建的。你看到的是带有 `OnItemClickListener` 的 `ListView` （指的是 `activity_reminders.xml` 里的 `reminders_list_view` ），（译者：在它的 `onItemClick` 方法里）创建了另一个 `modeListView` 还有另一个嵌入的 `OnItemClickListener` 来响应来自 `modeListView` 的选项点击事件。

内嵌的点击监听器弹出一个吐司信息来指示是编辑还是删除所点击的条目。它也重命名来自外部调用者 `OnItemClickListener` 的位置参数叫做 `masterListPosition` 来区分内置的 `OnItemClickListener` 的位置参数。这个 `masterListPosition` 用于吐司指明哪条提醒被用于可能的编辑或删除。最后， `dialog.dismiss()` 方法调用于点击监听器，用来完全移除对话框。

运行一下来测试新特性如图6-4所示。点击一条提醒接着在弹出对话框再点编辑或删除。如果吐司里报告的提醒位置和你点击的对不上，再确认一下你吐司里追加 `masterListPosition` 值而不是 `postion` 。按 `Ctrl+K | Cmd+K` 提交更改到Git，并附上信息Adds a ListView dialog for individual list items。

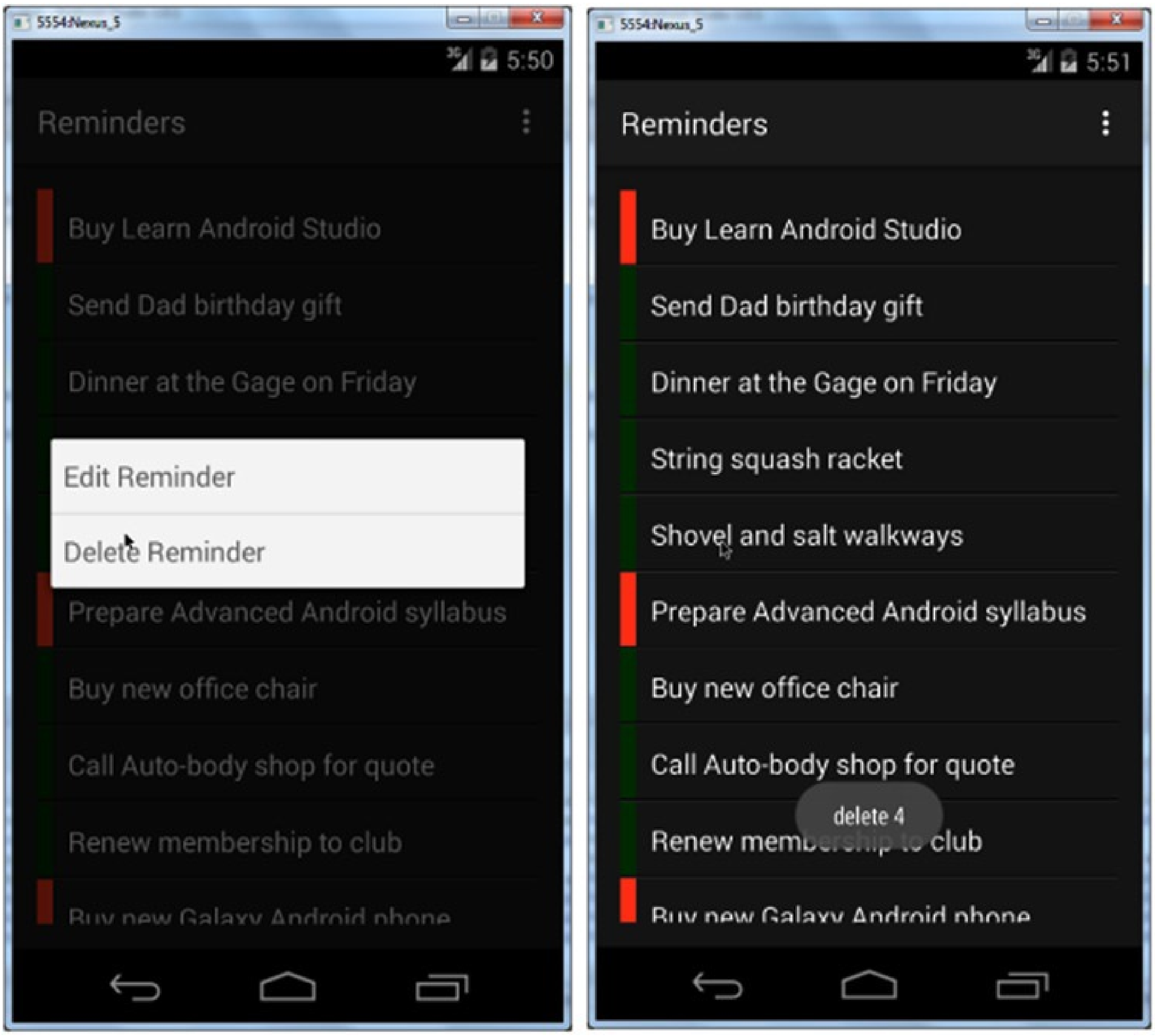


图6-4 模拟删除一条提醒

提供多选上下文菜单

随着这个app逐渐成形，现在将达成这种特性：在一次操作里允许多选提醒条目用于编辑。这种特性只能在API 11或更高版本上才有效。你将通过使用资源载入协定来有条件地达成这种特性。这个处理将在这章稍后解释并且第8章有所有的细节。你也将需要包含一个运行时检查看看是否支持这种特性。

为提醒行条目创建另一个备用的布局。打开项目工具窗口在资源夹上右击带出来的上下文。选择新的Android资源文件命名为r `eminders_row` ，如图6-5示。

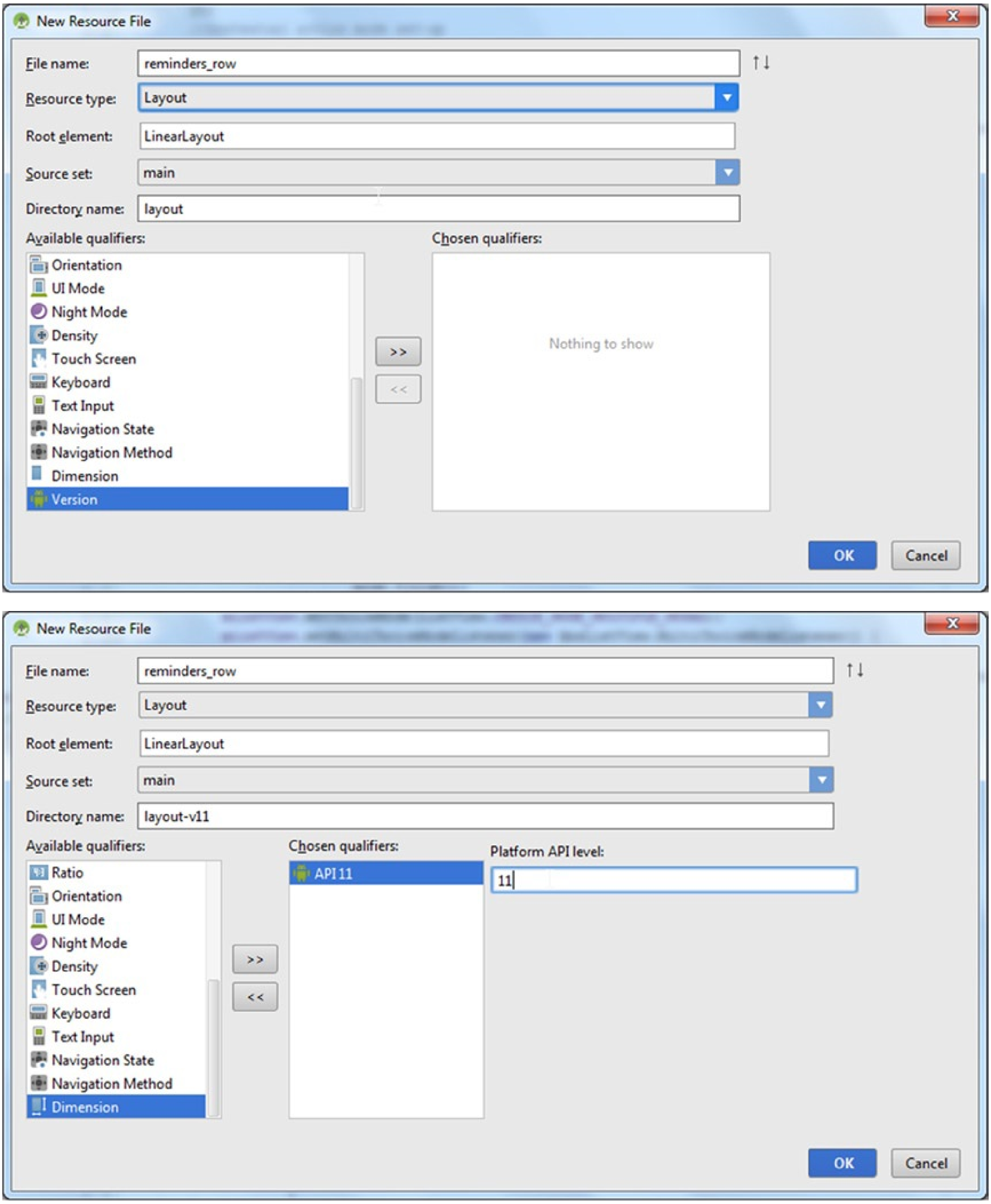


图6-5 新资源文件 `reminders_row`

选择资源类型为布局，这样就自动改变目录名为 `layout` 。在有效限定词部分选择相应的版本然后双击(`>>`)双V纹章按钮增加到选择的限定词里。输入11作为平台API级别并注意到目录名更新了，并反映了选择的限定词版本。那叫资源限定并且它们整合于运行时里，让你可以为特别的设备和平台版本定制你的用户界面。按回车（或点OK）接受这个新资源限定目录并继

续。如果你打开项目工具窗并设为Android示图，如图6-6，你将看到 `layout` 文件夹下所有的 `reminders_row` 布局文件在一起。还有，Android示图的项目工具窗让相关联的文件聚集在一起使你有效地管理它们。

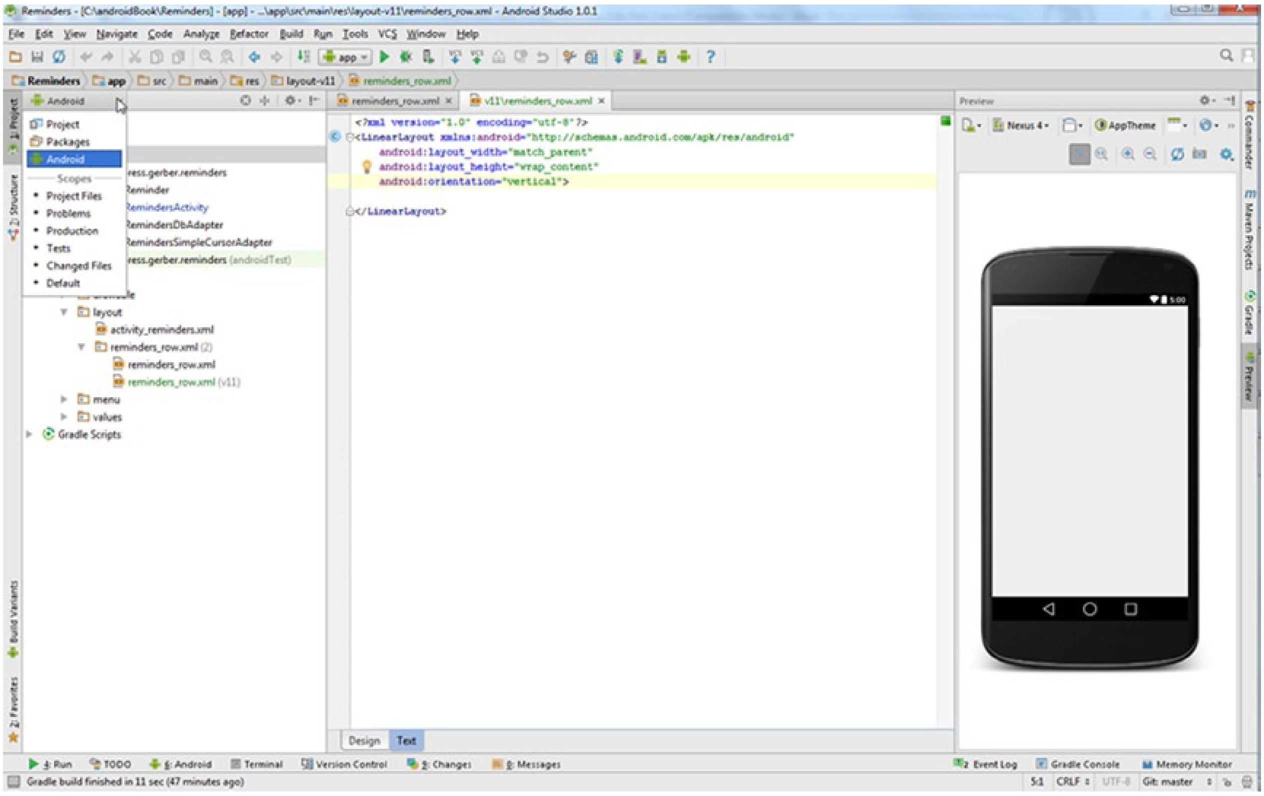


图6-6 聚集的布局

复制整个原始的 `reminders_row` 布局并粘贴到刚新建版本11的布局里。现在修改内层的水平线性布局的背景色属性如下：

```
android:background="?android:attr/activatedBackgroundIndicator"
```

这个值分配背景色属性带前缀 `?android:attr/` ，这是参考定义于Android SDK里的一种风格。Android SDK提供了很多这样的预定义属性，并且你可以使用它们到你的app上。在多选模式时 `activatedBackgroundIndicator` 属性使用了系统定义的有效背景色。

目标于早期的SDK

现在你将学习如何引入一个平台依赖的特性。打开项目工具窗并打开在Gradle剧本区下面的app模块 `build.gradle` 文件（它会在第二个入口里）。Gradle文件含有编译的构建逻辑和app的包装。参照你的app所支持的平台所有配置存在于这些特别的文件里（第十三章深度探索了Gradle构建系统）。注意到最低SDK版本设置为8，这让你的app可以运行在99%的Android设备上。现在我们将创建的这个特性需要最低SDK版本是11。涵括这节的代码和特性将允许用户运行在SDK11或更高版本带来的更先进的特性，叫上下文动作模式。而且，低于SDK11的将不会有这个特性，但更重要的是，他们的app不会因此而崩溃。

加入上下文动作模式

接下来介绍的多选模式的上下文动作模式菜单，是一个动作清单可用于所有选择项的上下文。加载一个新的菜单资源，在 `res/menu` 目录上右击选择 `New ➤ Menu` 资源文件并命名为 `cam_menu` 。以下列代码清单完成它：

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android";>
    <item android:id="@+id/menu_item_delete_reminder"
    android:icon="@android:drawable/ic_menu_delete"
    android:title="delete" />
</menu>
```

这个资源文件为上下文菜单定义了唯一的删除动作条目。这里你也正用了一点不同的属性值。这个特别的属性象之前的背景色属性一样也是访问Android内建的缺省值。不管怎样，在那前缀 `?android:attr/ prefix` 只用于参考风格属性。在这用在属性上的语法参照一个稍有不同的格式。用at符号 `@` 触发一个到资源值查询的命名空间。你可以用这种方式访问变量命名空间。Android命名空间是所有内建Android值所在的地方。用这个命名空间是多种变量资源所存在的如 `drawable` , `string` 以及布局。当你用了 `@+id` 为前缀，它会创建一个新的 `ID` 在你的项目的 `R.java` 文件里，且当你用到 `@id` 前缀，它会查找Andriod SDK的 `R.java` 文件存在的 `ID` 。这个例子里定义了一个新的 `ID` 名， `menu_item_delete_reminder` ，它结合于菜单选项。它也从 `android:drawable` 命名空间里拉出一个图标，作为它的图标。

用新的上下文菜单和一个备用的布局运行于API 11或更高版本，你可以加载一个有条件的检查以允许带有上下文动作菜单的多选模式。打开 `RemindersActivity` 并加上下面的代码到 `onCreate` 方法的后面。

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {

}
```

构建类从 `android.os` 包里导入并且给你访问一系列的常量值，那些可用于以指定的API级别来匹配设备。在这个例子里，你期望API级别等于或高于HONEYCOMB而它实际包含整数11。把清单6-4的代码插入到刚定义的块里。IF块保护了运行OS低于HONEYCOMB的系统不让这个app崩溃。

Listing 6-4. `MultiChoiceModeListener` Example

```
mListView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE_MODAL);
mListView.setMultiChoiceModeListener(new AbsListView.MultiChoiceModeListener() {
    @Override
    public void onItemCheckedStateChanged(ActionMode mode, int position, long id, bool
ean
    checked) { }
    @Override
    public boolean onCreateActionMode(ActionMode mode, Menu menu) {
        MenuInflater inflater = mode.getMenuInflater();
        inflater.inflate(R.menu.cam_menu, menu);
        return true;
    }
    @Override
    public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
            return false;
    }
    @Override
    public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
        switch (item.getItemId()) {
            case R.id.menu_item_delete_reminder:
                for (int nC = mCursorAdapter.getCount() - 1; nC >= 0; nC--) {
                    if (mListView.isItemChecked(nC)) {
                        mDbAdapter.deleteReminderById(getIdFromPosition(nC));
                    }
                }
                mode.finish();
                mCursorAdapter.changeCursor(mDbAdapter.fetchAllReminders());
                return true;
        }
        return false;
    }
    @Override
    public void onDestroyActionMode(ActionMode mode) { }
});
```

解决所有导入的问题。你将注意到 `getIdFromPosition()` 没有定义并标为红色（错误）。把光标放在这个方法上并按劳取酬 `Alt_Enter` 调用 `IntelliSense` 并选择创建方法。选择 `RemindersActivity` 作为目标类。选择整型作为返回类型。用清单6-5的代码完成这个方法。

Listing 6-5. `getIdFromPosition()` method

```
private int getIdFromPosition(int nC) {
    return (int)mCursorAdapter.getItemId(nC);
}
```

此处理逻辑定义了一个 `MultiChoiceModeListener` 并把它附到L `istView` 。无论何时你长按ListView的一个条目时，运行时调用 `onCreateActionMode()` 方法在 `MultiChoiceModeListener` 上。如果方法返回真值，就进入多选动作模式。这里重写方法的逻辑填充了一个上下文菜单用于显示动作条。使用多选动作模式的好处是你可以选择多行。一次点击选中了某个条目，接下来再点击则会去选这个条目。当你点击上下文菜单里的每个条目，运行时将带着被点的条目调用 `onActionItemClicked()` 方法。

在这个方法里，通过比较 `itemId` 和你加到菜单项里的删除元素的 `id` ，检查下删除条目是否被点击了。（看看在本节开始时所描述有关删除条目 `ID` 的XML清单）。如果这个条目被选，轮询所有的清单条目并要求 `mDbAdapter` 来删除它们。在删除所选条目后，逻辑调用动作模式

对象`finish()`方法，这将禁止多选动作模式并返回`ListView`到普通模式。接下来你调用`fetchAllReminders()`方法众数据库重新装载所有的提醒条并传送在`mCursorAdapter`对象`changeCursor`方法所返回的游标。最后，方法返回真值来表明动作被正确地处理了。所有其它的没被正确处理的逻辑，方法返回假，表明一些其它的事件监听器可能处理了这个点击事件。

Android Studio将高亮一堆错误语句，因为你正使用的API无效或低于Honeycomb。这外错误生成于Lint，一个状态分析工具内建于Android SDK并完全整合在Android Studio里。你需要加上下面的声明在`RemindersActivity.onCreate()`方法里，在`@Override`声明之上或之下都行，而且要目标API解决导入问题：

```
@TargetApi(Build.VERSION_CODES.HONEYCOMB)
```

这个特别的声明告诉Lint欺骗方法调用是提供的API级别的目标，不管构建配置所指定的。提交更改到Git用Adds Contextual Action Mode with context action menu作注释。图6-7描写了你可能看到的新特性。

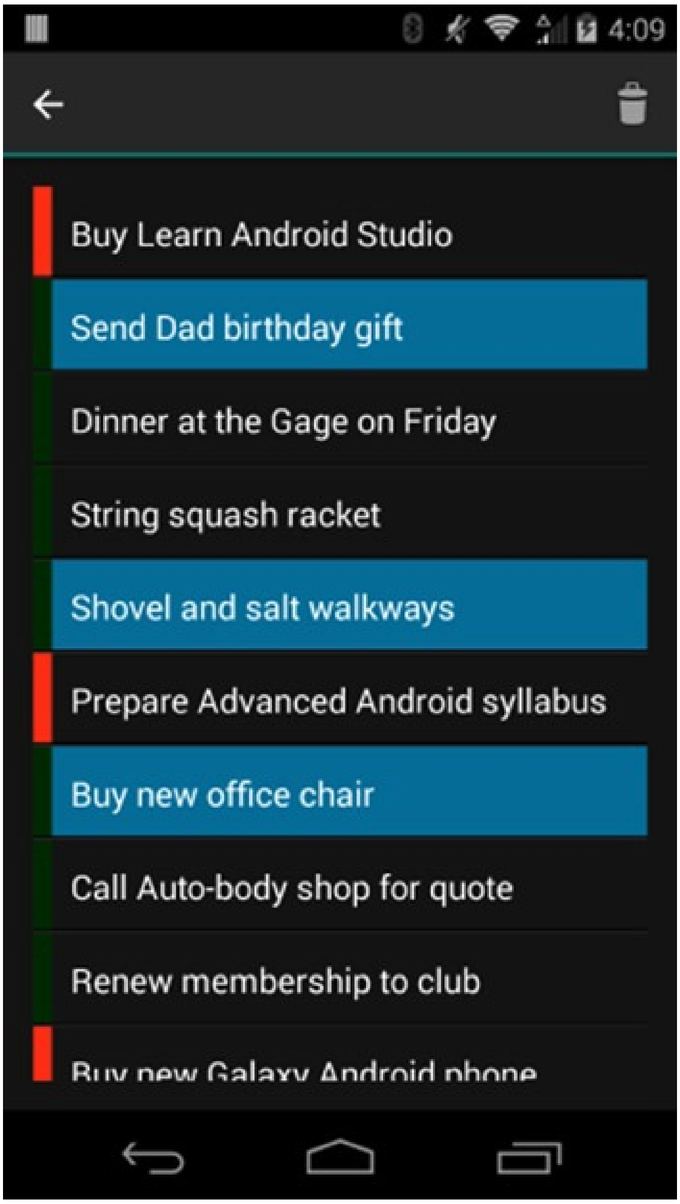


图6-7 允许多选模式

Implementing Add, Edit, and Delete

迄今为止，你已经添加了从清单里删除提醒的逻辑。这个逻辑在上下文动作模式里也可以有效执行。目前还无法插入新提醒或编辑现有的提醒。无论如何，你将创建一个用户自定义对话框或添加提醒，另一个用来编辑现有的提醒。最终，你将绑定这些对话框到 `RemindersDbAdapter` 。

在处理这个之前，先添加一些新的颜色。把这它们加到 `colors.xml` 文件里：

```
<color name="light_grey">#bababa</color>
<color name="black">#000000</color>
<color name="blue">#ff1118ff</color>
```

注意：通常，你的app可能有一个全面的颜色风格，这将保证所有屏幕和对话框的一致性。无论如何，颜色风格超出了这个简单例子的范围了。

策划一个用户对话框

开发的一个好习惯是：以简单工具优于执行它来描绘你的用户界面。这样在引入任何代码之胺，让你图形化元素如何适合屏幕。你可用一个编辑器如inkscape，它是跨平台的，或者你可以用笔记本的纸张和铅笔。在移动商务里，这些描绘称作线框图(wireframe)。

图6-8是我们的用户对话框的插图，完成于inkscape。这个线框图有意不正式，来强调元素的摆放好过一个精准的外观和感觉。

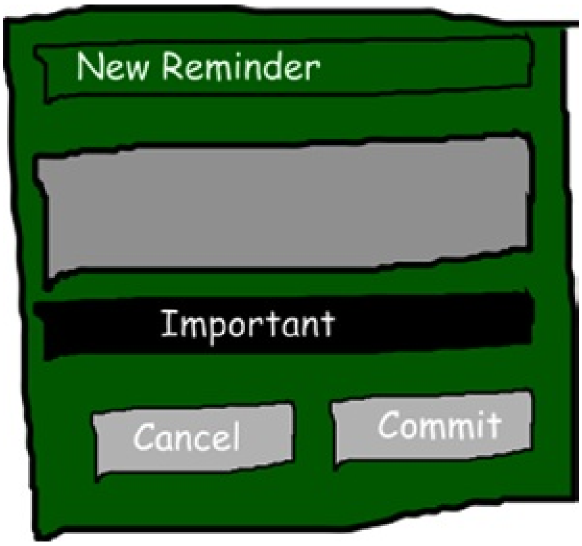


图6-8 线框图描绘用户自定义对话框

注意：这本书里的一些用户自定义的绘图和线框图正是用Inkspace创建的，一个多平台向量图形编辑器。在`www.inkscape.org`上，它是免费的。

把线框图就在那了，你可以开始计划如何排列屏幕的元素。因为大多数元素从上排到下，在最外层用一个竖直线性布局，这是最显然的选择。不管怎样，底下的两个按钮并排在一起。这样你可以用一个水平线性布局放在前面的竖直线性布局里。图6-9加上一些声明来描绘和高亮这些内嵌的元件。

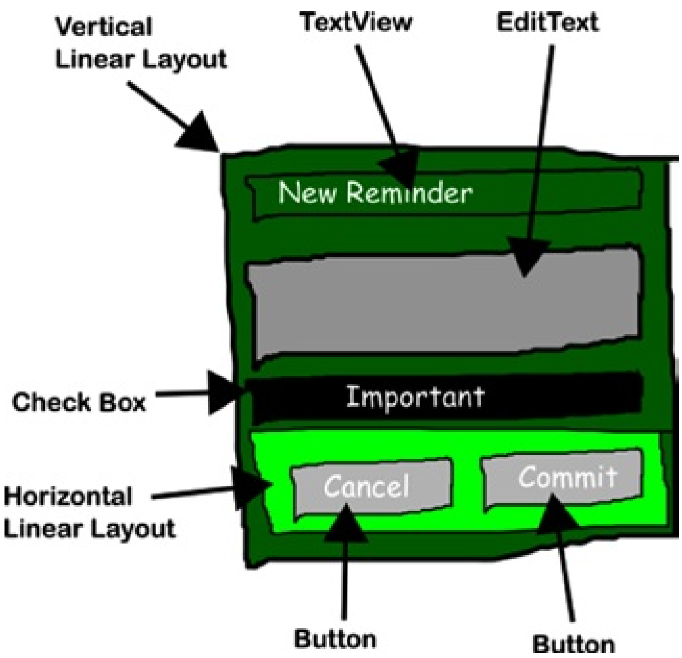


图6-9 线框图描绘 `widget` 标签

从策划到代码

随着线框图在那，尽量用图形化设计器来开发布局。开始在res目录上右击并选择创建一个新的Android资源文件然后命名为 `dialog_custom` ，资源类型为布局资源。以线性布局作为根元素，完成对话框。接着我们的线框图，从调色板拖放视图控件到这个平台上。清单6-6包含了这个完成的XML布局文件，并带有将在代码中用到的 `ID` 值。

Listing 6-6. Completed `dialog_custom.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/custom_root_layout"
android:layout_width="300dp"
android:layout_height="fill_parent"
android:background="@color/green"
android:orientation="vertical"
>
    <TextView
    android:id="@+id/custom_title"
    android:layout_width="fill_parent"
    android:layout_height="60dp"
    android:gravity="center_vertical"
    android:padding="10dp"
    android:text="New Reminder:"
    android:textColor="@color/white"
    android:textSize="24sp" />
    <EditText
    android:id="@+id/custom_edit_reminder"
    android:layout_width="fill_parent"
    android:layout_height="100dp"
    android:layout_margin="4dp"
    android:background="@color/light_grey"
    android:gravity="start"
    android:textColor="@color/black">
    <requestFocus />
    </EditText>
    <CheckBox
    android:id="@+id/custom_check_box"
    android:layout_width="fill_parent"
    android:layout_height="30dp"
    android:layout_margin="4dp"
    android:background="@color/black"
    android:paddingLeft="32dp"
    android:text="Important"
    android:textColor="@color/white" />
    <LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal">
        <Button
        android:id="@+id/custom_button_cancel"
        android:layout_width="0dp"
        android:layout_height="60dp"
        android:layout_weight="50"
        android:text="Cancel"
        android:textColor="@color/white"
        />
        <Button
        android:id="@+id/custom_button_commit"
        android:layout_width="0dp"
        android:layout_height="60dp"
        android:layout_weight="50"
        android:text="Commit"
        android:textColor="@color/white"
        />
    </LinearLayout>
</LinearLayout>
```

生成一个用户对话框

现在 `RemindersActivity` 里用完成的对话框布局了。清单6-7实现了一个新的 `fireCcustomDialog()` 方法。把这些代码放到 `RemindersActivit` 文件里去，仅在 `onCreateOptonsMenu()` 方法上面，并解决好导入的问题。

Listing 6-7. The `fireCustomDialog( )` Method

```
private void fireCustomDialog(final Reminder reminder){
    // custom dialog
    final Dialog dialog = new Dialog(this);
    dialog.requestWindowFeature(Window.FEATURE_NO_TITLE);
    dialog.setContentView(R.layout.dialog_custom);
    TextView titleView = (TextView) dialog.findViewById(R.id.custom_title);
    final EditText editCustom = (EditText) dialog.findViewById(R.id.custom_edit_remind
er);
    Button commitButton = (Button) dialog.findViewById(R.id.custom_button_commit);
    final CheckBox checkBox = (CheckBox) dialog.findViewById(R.id.custom_check_box);
    LinearLayout rootLayout = (LinearLayout) dialog.findViewById(R.id.custom_root_layo
ut);
    final boolean isEditOperation = (reminder != null);
    //this is for an edit
    if (isEditOperation){
        titleView.setText("Edit Reminder");
        checkBox.setChecked(reminder.getImportant() == 1);
        editCustom.setText(reminder.getContent());
        rootLayout.setBackgroundColor(getResources().getColor(R.color.blue));
    }
    commitButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            String reminderText = editCustom.getText().toString();
            if (isEditOperation) {
                Reminder reminderEdited = new Reminder(reminder.getId(),
                reminderText, checkBox.isChecked() ? 1 : 0);
                mDbAdapter.updateReminder(reminderEdited);
                //this is for new reminder
            } else {
                mDbAdapter.createReminder(reminderText, checkBox.isChecked());
            }
            mCursorAdapter.changeCursor(mDbAdapter.fetchAllReminders());
            dialog.dismiss();
        }
    });
    Button buttonCancel = (Button) dialog.findViewById(R.id.custom_button_cancel);
    buttonCancel.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            dialog.dismiss();
        }
    });
    dialog.show();
}
```

`fireCustomDialog( )` 方法将用于插入和编辑，就算它们间有少许不同。方法的开始三行创建了一个Android对话框，没有标题并填充清单6-6的布局。 `fireCustomDialog( )` 方法接着从这个布局里查找所有重要的元素并储存到本地变量里。然后方法查看reminder参数是否为空，设置一个 `isEditOperation` 布尔变量。如果说有一个提醒被传入（或者值为不空），方法会假设这不是一个编辑操作并把这个变量设为假；否则，设为真。如果调用 `fireCustomDialog( )` 方法的是一个编辑操作，标题设为 `Edit Reminder` 而且 `CheckBox` 和 `EditText` 用提醒参数的值来设定。这个方法也设置了最外层容器布局的背景色为蓝色，为了从视觉上区别插入对话框和编辑对话框。

接下来的几行由一个代码块组成，为提交按钮设置及定义一个 `OnClickListener` 。这个监听器响应点击事件，更新数据库。再次， `isEditOperation()` 被检查了，如果编辑操作进行中，那么一个新提醒创建了，用来自提醒参数的 `ID` 以及 `EditTex` 的值还有在屏的复选框值。这个提

醒传送给 `mDbAdapter` ，通过 `updateReminder()` 方法。

如果编辑没有进行，这个逻辑查询 `mDbAdapter` 来创建一个新的提醒到数据库，用 `EditText` 的值还有在屏的复选框值。在无论是更新或创建调用后，所有提醒通过 `mCursorAdapter.changeCursor()` 方法会重新装载.这个逻辑有点象之前添加的清单6-5代码。点击监听器在提醒重新装载后解散对话框。

在配置提交按钮的点击行为后，这个例子设置了另一个取消按钮的监听器。这个监听器只是简单地解散对话框。所有按钮的行为定义后，这个例子结束于显示用户对话框。

现在你可以为在 `onCreate()` 方法里的 `modeListView` 用这个新方法在 `OnItemClickListener` 中。为这个监听器找到 `onItemClick()` 方法用下面的代码替换原来所有代码。

```
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    //edit reminder
    if (position == 0) {
        int nId = getIdFromPosition(masterListPosition);
        Reminder reminder = mDbAdapter.fetchReminderById(nId);
        fireCustomDialog(reminder);
        //delete reminder
        } else {
        mDbAdapter.deleteReminderById(getIdFromPosition(masterListPosition));
        mCursorAdapter.changeCursor(mDbAdapter.fetchAllReminders());
    }
    dialog.dismiss();
}
```

为了编辑提醒，去掉 `Toast.makeText()` 调用，替换为用 `ListView` 位置来找到提醒的调用。这价目提醒然后被传送到 `fireCustomDialog()` 方法来触发编辑行为。为删除提醒，用和清单6-5一样的代码逻辑。再次， `mDbAdapter.deleteReminderById()` 用于删除提醒，并且 `changeCursor()` 用于从 `mDbAdapter.fetchAllReminders()` 调用中返回游标。

找到 `onOptionsItemSelected()` 方法，在 `RemindersActivity.java` 文件很底部的地方，编辑它如清单6-8。

Listing 6-8. `onOptionsItemSelected` Definition

```
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_new:
            //create new Reminder
            fireCustomDialog(null);
            return true;
        case R.id.action_exit:
            finish();
            return true;
        default:
            return false;
    }
}
```

这里，当你选择了菜单项是`action_new`，简单地调用了`fireCustomDialog()`。传送一个空值给它，象之前提及的空值检查并设置`isEditOperation`为假，然后征用一个新提醒对话框。运行app检查下新特性。你会看到新的用户对话框。当创建新提醒时，你将看到一个绿色对话框，而编辑提醒时是一个蓝色的对话框，如图6-10和6-11所分别对应的。测试菜单项确保创建和删除操作功能象它们应该的那样。提交更改到Git，附上提交信息：Adds database Create, Read, Update, and Delete support with custom dialogs。

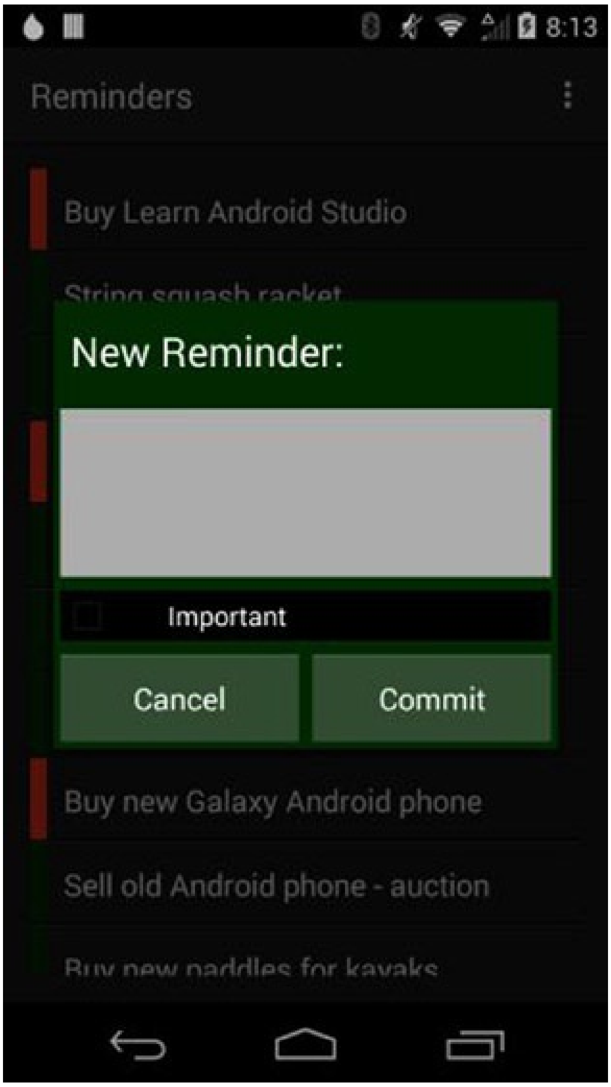


图6-10 新提醒对话框

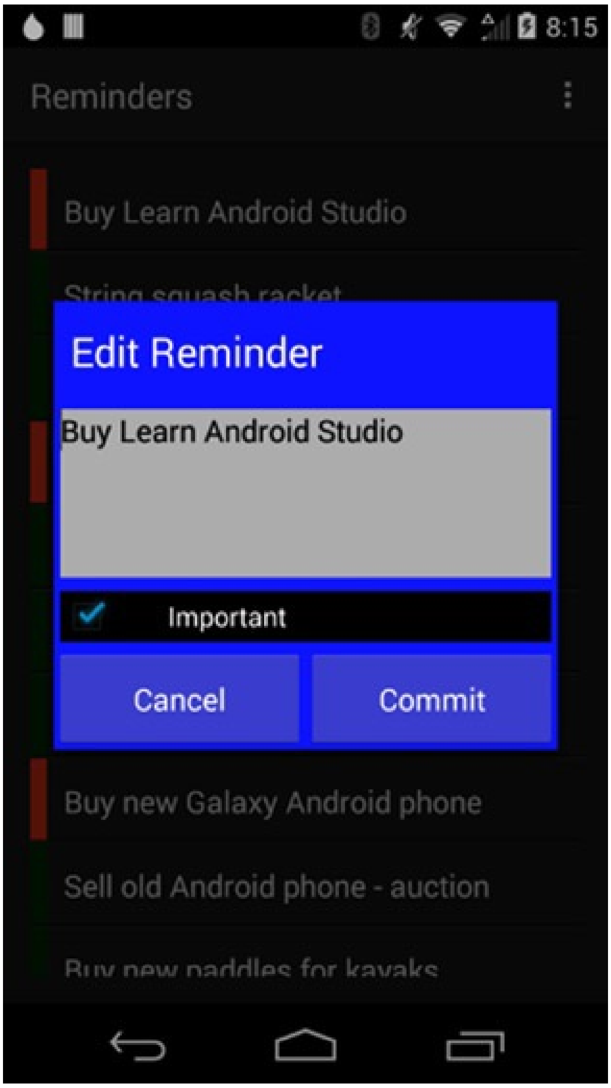


图6-11 编辑提醒对话框

加上用户图标

所有的特性都放置好了，你可以加上一个用户图标作为完成点缀。可以用任何图形编辑器来创建它，或者如果你不喜欢图形编辑，从网上下点免费的。我们的例子放置ic_launcher图标，它创建于Inkscape。找开项目工具窗并右击 `res/mipmap` 目录。选 `New ‣ Image Asset` 。将看到图6-12那样的对话框。点击 `Image File:` 选择框最右边的省略号图标，导航到你要的图片位置并插入进来。让其他的设置如图6-13所示。点下一步，接着点完成。

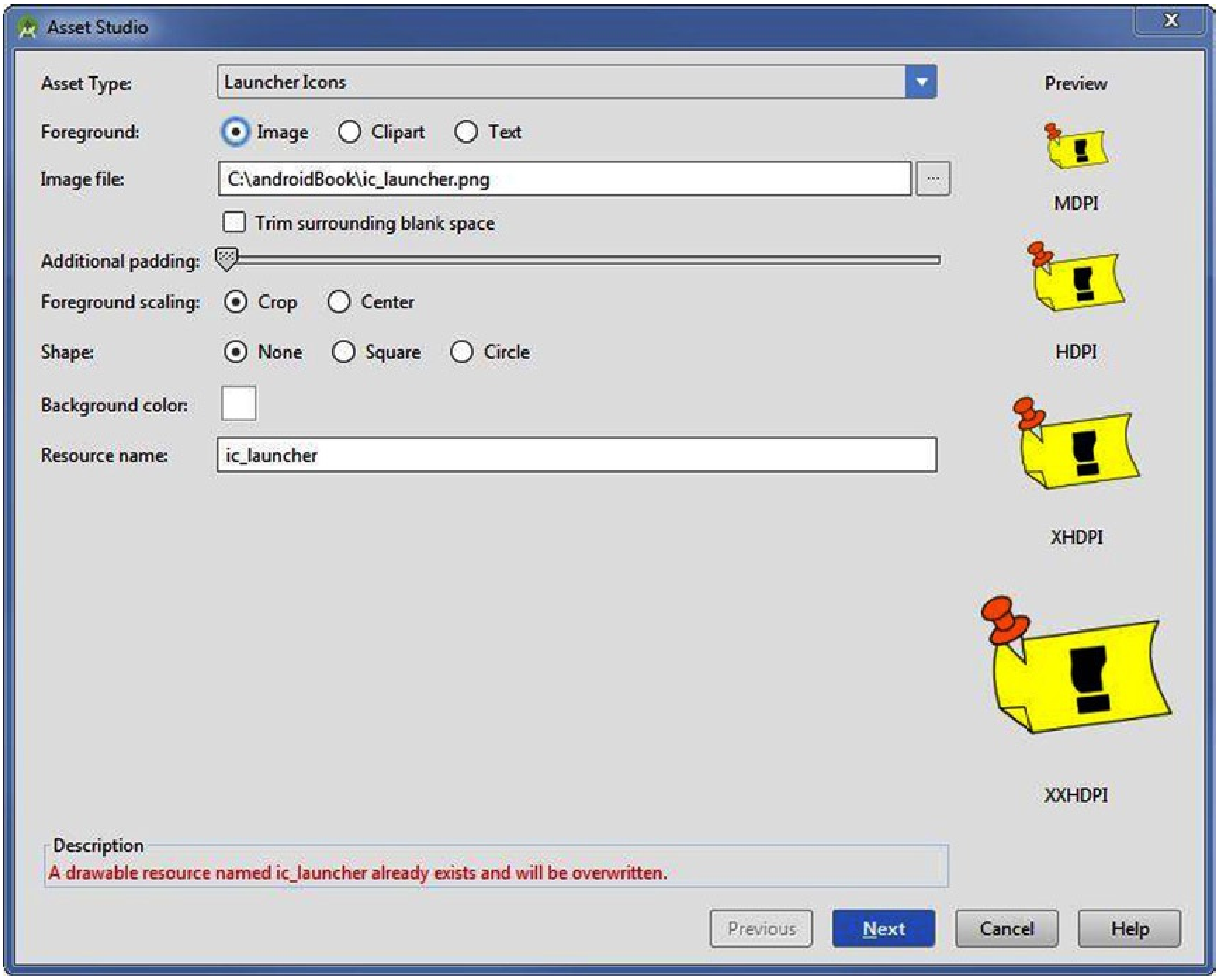


图6-12 新图片资源对话框

有不少目录使用了 `mipmap` 这个名字。这些目录每个都有着后缀，那个指派屏幕尺寸限定。Android运行时将从特别的目录里拉出资源，依照app所运行设备屏幕的解析度。资源夹和它们的后缀在第8章将有详解。

插入下面的几行代码到 `onCreate()` 方法里，在填充布局之后，即 `setContentView(R.layout.activity_reminders);` 。这些代码将显示用户图标在你的动作条里：

```
ActionBar actionBar = getSupportActionBar();
actionBar.setHomeButtonEnabled(true);
actionBar.setDisplayShowHomeEnabled(true);
actionBar.setIcon(R.mipmap.ic_launcher);
```

运行代码，将看到用户图标在动作条里。图6-13展示了以用户图标运行的app。

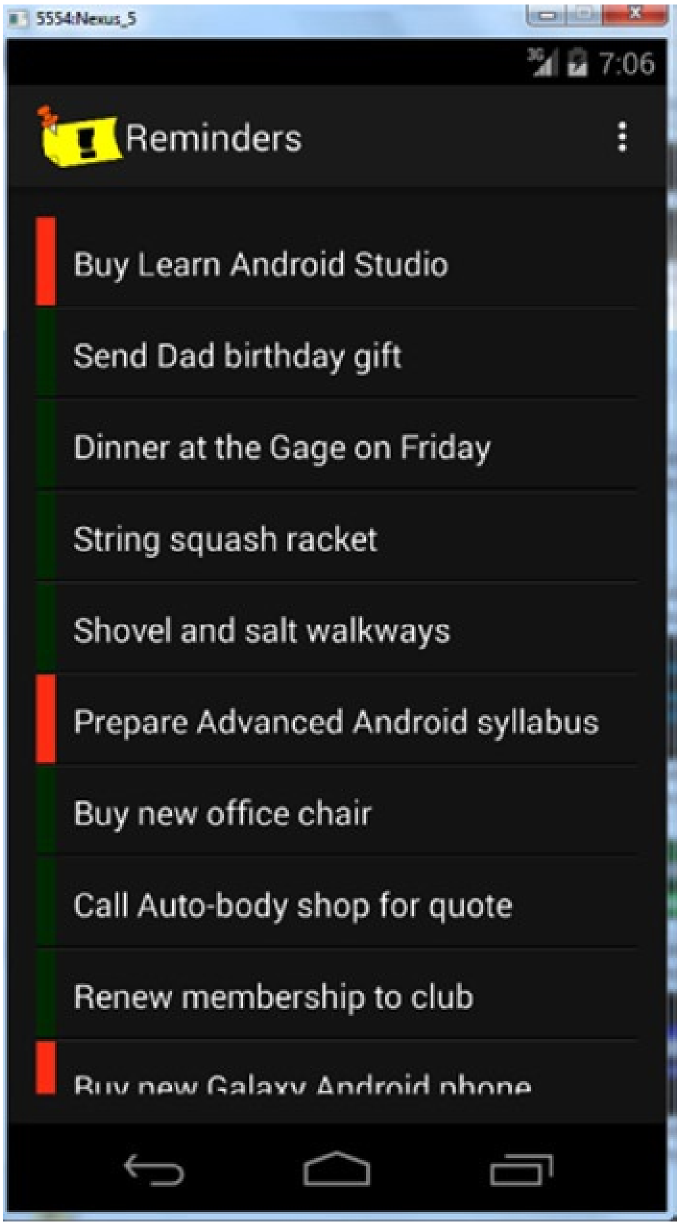


图6-13 动作条的用户图标

按 `Ctrl+K | Cmd+K` 提交更改到Git，以Adds a custom icon作为提交信息。

小结

恭喜！你已实现了使用Android Studio完成的第一个Android app。在这个过程里，你学会了用图形化设计器编辑XML布局文件。你也学会了如何用文本模式编辑一个行XML布局。这章展示了在支持的平台上如何有条件地实现上下文动作模式。最后，你看到了如何加载一个用户图标到不同的分辨率的屏幕上。